Ensemble Methods for Native Language Identification

Sophia Chan,[*] Maryam Honari Jahromi,[*] Benjamin Benetti,[*] Aazim Lakhani, Alona Fyshe
Department of Computer Science
University of Victoria
Victoria, BC, Canada
{schan1,mhonari,bbenetti,aazimlakhani,afyshe}@uvic.ca

Abstract

Our team—Uvic-NLP—explored and evaluated a variety of lexical features for Native Language Identification (NLI) within the framework of ensemble methods. Using a subset of the highest-performing features, we train Support Vector Machines (SVM) and Fully Connected Neural Networks (FCNN) as base classifiers, and test different methods for combining their outputs. Restricting our scope to the closed essay track in the NLI Shared Task 2017, we find that our best SVM ensemble achieves an F1 score of 0.8730 on the test set.

1 Introduction

Native Language Identification (NLI) is the task of identifying a person's native language (L1) based on a sample of their writing or speech in a second language (L2). The underlying intuition is that those with the same L1 tend to use similar language patterns during L2 production. This is known as cross-linguistic influence (Ortega, 2014).

NLI can accelerate second language acquisition by giving students L1-specific feedback on their written or spoken samples (Malmasi et al., 2014). In forensic linguistics, NLI can be applied to identify the L1 of anonymous texts (Perkins, 2015).

The NLI Shared Task 2013—the first of its kind—was based on written essays (Tetreault et al., 2013), while the 2016 Computational Paralinguistics Challenge was based on spoken responses (Schuller et al., 2016). The NLI Shared Task 2017 organizers provided a dataset of both essays and transcriptions of verbal responses (Malmasi et al., 2017). As our team—Uvic-NLP—participated in the closed essay track, we performed classification on essays only.

We begin our analysis by comparing various lexical features and focus on two high-performing classifiers: Support Vector Machines (SVM) and Fully Connected Neural Networks (FCNN). Then, we explore different ensemble methods for combining outputs of individual classifiers. We present and discuss three of our best systems for this task: a single SVM classifier, an SVM ensemble, and an FCNN ensemble.

2 Related Work

NLI is generally conceptualized as a multi-class supervised classification problem, where the classes represent the set of possible L1s. One of the first NLI systems trained SVMs on a variety of stylistic features (Koppel et al., 2005).

The NLI Shared Task 2013 introduced a corpus designed specifically for NLI (Blanchard et al., 2013). Use of a standardized dataset and evaluation metric allowed for the effective comparison of different models, and the results confirmed the usefulness of SVMs for NLI (Tetreault et al., 2013). Popular features included word, part of speech (POS), and character *n*-grams; higher-order *n*-grams were shown to be especially useful. Four of the top five teams used at least 4-grams, with the top team using up to 9-grams. String kernels using 5- to 8-grams at the character-level also worked well, and were one of the best performing models for this task (Ionescu et al., 2014).

A trend in recent work is the use of ensemble methods, which combine the predictions of a set of classifiers, giving more accurate results than a single classifier trained on a combination of different features (Tetreault et al., 2012; Malmasi et al., 2013). Malmasi and Dras (2017) used meta-

[*] These authors contributed equally to this work.

classifier ensembles, where results from base classifiers are fed to an ensemble of meta-classifiers. Such models are the current state of the art for NLI.

3 Data

The dataset for the essay track of the NLI Shared Task 2017 was collected by Educational Testing Services, and consists of written responses to a standardized assessment of English proficiency for academic purposes.

13,200 response essays from test takers were separated into three sets: 11,000 for training (TRAIN), 1,100 for development (DEV), and 1,100 for testing (TEST). Each set of documents is equally distributed among eleven L1s: Arabic (ARA), Chinese (CHI), French (FRE), German (GER), Hindi (HIN), Italian (ITA), Japanese (JPN), Korean (KOR), Spanish (SPA), Telugu (TEL), and Turkish (TUR).

4 Features

Previous work demonstrates that a variety of lexical and syntactic features are useful for NLI (Tetreault et al., 2012). In addition to incorporating lexical features known to be effective for this task, we also extract phonemes. Here, we describe each of the features in turn.

Word *n*-grams Where topic bias is pervasive, word *n*-grams are not useful features for classification (Brooke and Hirst, 2011), but have been used successfully in topic-balanced corpora (Tetreault et al., 2012). Our dataset is balanced across topics, making word *n*-grams useful.

Lemma *n*-grams Lemmas are the dictionary representation of words, i.e. words that are stripped of morphological marking. The lemmatized versions of all words in our corpus were attained using Natural Language Toolkit's WordNet interface (Bird et al., 2009; Feinerer and Hornik, 2016; Wallace, 2007; Fellbaum, 1998).

Character *n*-grams Tsur and Rappoport (2007) achieved good results on the NLI task using only character bigrams as features. Methods working at the character level were also the previous state of the art (Ionescu et al., 2014). Character *n*-grams can be generated from text within or across word boundaries.

Part of speech *n*-grams Koppel et al. (2005) found rare part of speech (POS) bigrams to be a useful feature; many teams in the 2013 Shared Task also made use of this feature (Tetreault et al., 2013). We use the Stanford Tagger to extract POS features (Toutanova et al., 2003).

Function words Function words are a closed class of words that serve a grammatical function in sentences, whose use for NLI was explored early on (Koppel et al., 2005). These include articles, determiners, conjunctions, and auxiliaries. These were extracted based on a list provided in the ModErn Text Analysis Toolkit (Massung et al., 2016).

Spelling errors Spelling errors were extracted by finding the difference between misspelled words before and after they were corrected using the autocorrect package (Jonas, 2013). We coded a subset of the spelling errors defined by Koppel et al. (2005): repeated letter, double letter appears only once, letter replacement, letter inversion, inserted letter, and missing letter.

Phoneme *n*-grams Phonemes are representations of sounds in a language. In English, one sound can be represented using many different letters (e.g. *c*at and *k*ick). For mapping orthography onto phonemes, we used the Carnegie Mellon Pronouncing Dictionary (Weide, 2005). To our knowledge, phonemes have not yet been explored as a feature.

5 Classifiers

We evaluated classifier performance across features types and found that the SVM and FCNN classifiers consistently outperformed other classifiers, such as Perceptron and Multinomial Naive Bayes. As such, we focus on these two classifiers in subsequent experiments.

Ensemble methods involve combining the outputs of multiple classifiers to yield a final prediction (Polikar, 2006). Three types of ensemble methods which have been shown to be useful for NLI are explored here (Malmasi and Dras, 2017). At a high level, SVM and FCNN outputs are combined using (1) a voting scheme, (2) a Linear Discriminant Analysis (LDA) classifier trained on the outputs, and (3) multiple LDA classifiers—trained on random subsets of the outputs—whose predictions are in turn combined using a voting scheme.

Table 1: Comparison of individual feature types using SVM and FCNN classifiers, using F1 scores on DEV. The highest F1 score for each feature set is indicated in bold.

Feature type	SVM	FCNN
Word unigram	0.6936	0.7645
Word bigram	0.7228	**0.8027**
Word trigram	0.6705	0.6790
Lemma	0.6703	**0.7481**
Character bigram	0.4787	0.5818
Character trigram	0.6360	0.7381
Character 4-gram	0.7213	0.7836
Character 5-gram	0.7363	**0.8081**
POS bigram	0.4286	0.4081
POS trigram	**0.4723**	0.4472
Function words	0.3036	**0.5646**
Spelling errors	0.2201	**0.2509**
Phoneme bigram	0.5356	0.5509
Phoneme trigram	0.6697	0.6654
Phoneme 4-gram	0.7089	0.6727
Phoneme 5-gram	**0.7241**	0.6654
Combined	**0.8183**	0.7784

5.1 Support Vector Machine (SVM)

SVMs (Joachims, 1998) are frequently used for text classification and have been applied successfully to NLI (Tetreault et al., 2013). We use a scikit-learn SVM implementation: LinearSVC (Pedregosa et al., 2011).

5.2 Neural Networks

Since we found little previous work applying neural networks to NLI, this paper strives to fill this gap by constructing a FCNN using TensorFlow (Allaire et al., 2016) and the Keras (Chollet et al., 2015) framework.

The network is comprised of one hidden layer of 128 nodes that uses a tanh activation function and an input dropout of 0.2. The optimal dropout value was established empirically. Following the hidden layer, there is an 11 node output layer that uses the softmax activation function. The entire network uses a cross entropy loss function and the Adam optimization algorithm.

Due to memory constraints, we limit analysis to only the 100,000 most important features, selected by performing an ANOVA F-test on the entire feature set (Harwell et al., 1992).

In addition to the FCNN, we test another type of neural network for this task. Following the architecture described by Wang et al. (2016), we train a pipeline consisting of a convolutional neural network (CNN) which transforms the input data at the character-level and a Long Term Short Memory (LSTM) neural network which performs classification on the output of the CNN. We also trained an LSTM on word vectors (Mikolov et al., 2013). In both cases, however, we found results to be lacking in accuracy.

5.3 Ensemble construction

For any given SVM or FCNN, the output for 11-way classification can be represented as a vector of 11 numbers. For the SVM, output is in the form of confidence scores for each class, which is equivalent to the signed distance of that sample to each class's hyperplane (Weston and Watkins, 1998). Similarly, each FCNN prediction is in the form of confidence values for each class, derived from the softmax output layer.

Using the best feature combination and representation from the previous experiments, we trained two sets of base classifiers—FCNNs and SVMs—on different features and combined each set of outputs using three different voting schemes (Polikar, 2006):

- **Mean**: Final label is the class corresponding to the greatest average confidence score.
- **Median**: Final label is the class corresponding to the greatest median confidence score.
- **Plurality vote**: Final label is the class with the greatest number of votes. In a tie, we choose the class that comes first alphabetically.

In line with previous work, we achieve the highest accuracy using the mean rule (Malmasi et al., 2013), as shown in Table 3.

5.4 Meta-classifier

Another way to combine the outputs of several base classifiers is to feed their outputs into another classifier, also known as a meta-classifier. To obtain outputs from SVMs and FCNNs, we split the training set into ten folds and perform cross-validation. This gave us a set of meta-features that were then used as input to an LDA meta-classifier, which was found to outperform other algorithms for meta-classification in Malmasi and Dras (2017).

5.5 Meta-classifier ensembles

Building on the idea of ensembles and meta-classification, we experiment with ensembles of meta-classifiers (Malmasi and Dras, 2017). SVM and FCNN outputs—meta-features—are generated in the same way as in section 5.4. However, instead of training a single meta-classifier on these features, we use bagging (bootstrap aggregating) to train multiple LDAs on random subsets of the base classifier outputs. A grid search was performed to find the optimal number of meta-classifiers and optimal percentage of samples to train each LDA on. The predictions from multiple LDAs were then combined using voting schemes described in section 5.3.

6 Results and Discussion

In this section we present our results on single features, feature combinations, single classifiers, and classifier ensembles.

6.1 Individual features

The results of SVM and FCNN classifiers trained on different features are shown in Table 1. For these experiments, features were represented by their frequency count. We observe a general trend within different feature types: F1 scores increase as *n*-gram order increases (see Table 1). This is not unexpected, given the success of NLI models that make use of higher-order *n*-grams (Jarvis et al., 2013; Tetreault et al., 2013). One exception to this trend is that there seems to be a upper-bound for word *n*-grams at the bigram level, where accuracy drops for word trigrams. This may be attributed in part to the increased sparsity of features when we move from bigrams to trigrams at the word-level.

Interestingly, spelling errors were less informative than what we had expected. Although we did not evaluate the accuracy of the autocorrect package we used for spelling correction, we suspect that it did not perform well since it operates naively, without looking at context (Jonas, 2013). Additionally, the types of errors we defined might have not been fine-grained enough to capture differences unique to groups of L1 writers.

6.2 Single classifier results

As in Malmasi et al. (2013), we measure the effectiveness of different feature representations. Of the feature types described above, we include in our final system only a subset of the highest performing features. Thus, analysis is limited to this subset of features.

With frequency counts as a baseline, we compare the performance of classifiers trained on three different combinations of high-performing features. These groups are:

- **Word**: Lemmas, words (1-, 2-, and 3-grams).
- **Char**: Characters (4- and 5-grams).
- **Phoneme**: Phonemes (4- and 5-grams).

Each group of features is tested with and without term frequency-inverse document frequency (TF-IDF) weighting. Further, we examine the effects of binarization, L1 normalization, and L2 normalization on the same feature set. Note that L1 and L2 normalization refer to the vector norms across each input row. These results are summarized in Table 2.

Comparing classifiers trained on individual features (Table 1) to those trained on combinations of features (Table 2), it is evident that better results are achieved by training a single classifier on multiple features than on any single feature type. Further, Table 2 shows that the best performing classifiers use L2-normalized features with TF-IDF.

Our official submission to the NLI Shared Task 2017 used a single SVM classifier, which requires less time and fewer computational resources to train compared to a FCNN. An SVM on words (1-, 2-, and 3-grams) and characters (4- and 5-grams) achieves an F1 score of 0.8633 on TEST (see Table 4). The features were binarized, L2-normalized and TF-IDF weighted. The confusion matrix is shown in Figure 1.

6.3 Ensembles

The results detailed in this section were not submitted as part of the NLI Shared Task 2017, and were obtained after the test phase ended.

At the most basic level, individual classifiers are combined in a straightforward manner using a voting scheme. As we increase the complexity of the model, first by training an LDA meta-classifier on

Table 2: Comparison of feature representations for SVM and FCNN classifiers, using F1 scores on DEV. The best feature representation for each classifier is indicated in bold.

		Word		**Char**		**Phoneme**	
		SVM	**FCNN**	**SVM**	**FCNN**	**SVM**	**FCNN**
TF	*Binarized*	0.7983	**0.8190**	0.6932	**0.8172**	0.7550	0.6950
	Frequency counts	0.8090	0.8090	0.6931	0.8003	0.7056	0.6971
	L1 Normalized	0.6167	0.6372	0.4921	0.4427	0.4270	0.4604
	L2 Normalized	0.7736	0.7854	0.7629	0.7610	0.7677	0.6971
TF-IDF	*Binarized*	**0.8092**	0.7872	0.6837	0.7693	0.7489	0.6623
	Frequency counts	0.7772	0.7579	0.6834	0.7560	0.7085	0.6578
	L1 Normalized	0.6954	0.7845	0.5911	0.3794	0.5325	0.2919
	L2 Normalized	0.8049	0.8155	**0.7709**	0.8048	**0.7812**	**0.7059**

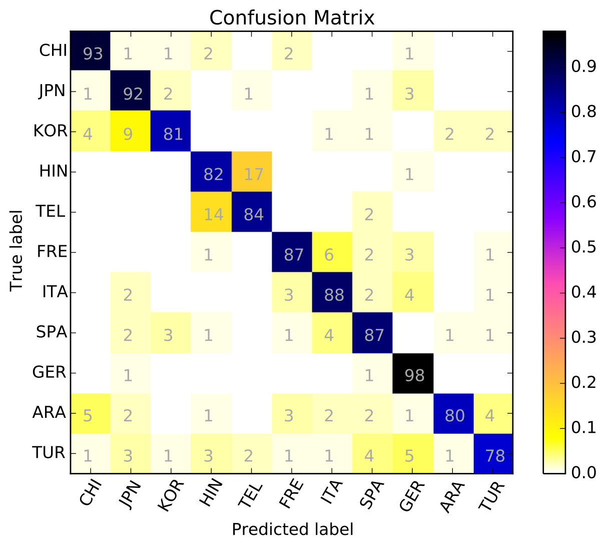

Figure 1: SVM confusion matrix on TEST. The SVM was trained on words (1-, 2-, and 3-grams) and characters within word boundaries (4- and 5-grams).

Table 3: Comparison of different ensemble methods to combine outputs of SVM and FCNN classifiers: voting schemes, LDA meta-classifier, and an ensemble of LDA meta-classifiers. F1 scores on DEV are shown. The best result for each classifier is indicated in bold.

		SVM	FCNN
Voting scheme	Plurality vote	0.8285	0.8109
	Mean	0.8417	0.8345
	Median	0.8313	0.8363
Meta-classifier	LDA	0.8448	0.8534
Meta-classifier ensembles	Plurality-LDA	0.8449	0.8507
	Mean-LDA	**0.8475**	**0.8544**
	Median-LDA	**0.8475**	**0.8544**

the outputs, and then by constructing an ensemble of meta-classifiers, we observe a slight performance gain for both SVMs and FCNNs at each step, consistent with the results in Malmasi and Dras (2017). Table 3 summarizes our results from using different ensemble methods to combine individual classifiers trained on words (2- and 3-grams), characters (4- and 5-grams) and phonemes (4- and 5-grams).

Further experiments with SVMs and FCNNs were conducted by selecting different features to combine on a trial and error basis. The decision to use character *n*-grams within as opposed to across word boundaries was made arbitrarily. All features are binarized, L2-normalized, and TF-IDF weighted. The results of our best ensemble classifiers on DEV and TEST are displayed in Table 4.

While an ensemble of meta-classifiers outperforms both a simple voting scheme and a single meta-classifier, we do not observe the same performance gain with respect to FCNNs (see Table 3).

Our best SVM ensemble consists of an ensemble of meta-classifiers. SVMs are trained on words (2- and 3-grams), characters within word boundaries (4- and 5-grams), and phonemes (4- and 5-grams), giving a total of six classifiers. The outputs of these individual classifiers are fed to an ensemble of LDAs, as described in 5.5. Finally, the LDA predictions are combined using the mean rule. The F1 score on TEST for this model is 0.8730.

Our best FCNN ensemble applies a voting scheme to classifier outputs. Four FCNN networks

Table 4: F1 scores on TEST and on DEV for final systems. Ensemble results were obtained after the test phase. The best result for each dataset is indicated in bold. * = Official submission to the NLI Shared Task 2017.

System	DEV	TEST
Random baseline	0.9090	—
Official baseline	0.7104	—
*SVM**	0.8168	0.8633
SVM ensemble	0.8475	**0.8730**
FCNN ensemble	0.8576	0.8560

are trained on the following combination of features: (1) word bigrams and lemma trigrams, (2) word bigrams, (3) character 5-grams, (4) character 5-grams within word boundaries. The outputs from these individual networks are combined using the mean rule, yielding an F1 score of 0.8560 on TEST.

Additionally, we created an ensemble of different SVM and FCNN classifiers but found no improvement over pure ensembles of either type.

7 Future work

We excluded from our system individual features that did not perform well in our experiments. It would be helpful to evaluate the influence of these less accurate features and determine whether they would be useful to include in ensemble classifiers. Further, we tested a limited number of combinations of features. One facet of the problem involves developing a systematic approach to search for a good feature set.

Although we trained several FCNNs on different feature types, its utility as a meta-classifier has not been examined.

A CNN-LSTM model shown to perform well for sentiment analysis (Wang et al., 2016) did not achieve good results for NLI. While sentiment classification typically involves five or fewer classes, there were 11 classes for the NLI Shared Task 2017. It may may be that additional classes increase the possibility of error. Further investigation is required to explain why a CNN-LSTM architecture performs worse relative to a FCNN model.

Our results show the utility of various features for this task and confirm that ensemble methods perform better than single classifiers trained on multiple features. They also offer several new directions to further improve NLI systems.

References

Joseph Allaire, Dirk Eddelbuettel, Nick Golding, and Yuan Tang. 2016. tensorflow: R interface to tensorflow. https://github.com/rstudio/tensorflow.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Daniel Blanchard, Joel Tetreault, Derrick Higgins, Aoife Cahill, and Martin Chodorow. 2013. Toefl11: A corpus of non-native english. Technical report, Educational Testing Service.

Julian Brooke and Graeme Hirst. 2011. Native language detection with 'cheap' learner corpora. In *Conference of Learner Corpus Research*. Presses universitaires de Louvain.

François Chollet et al. 2015. Keras. https://github.com/fchollet/keras.

Ingo Feinerer and Kurt Hornik. 2016. *wordnet: WordNet Interface*. R package version 0.1-11. https://CRAN.R-project.org/package=wordnet.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Michael Harwell, Elaine Rubinstein, William Hayes, and Corley Olds. 1992. Summarizing monte carlo results in methodological research: The one-and two-factor fixed effects ANOVA cases. *Journal of Educational Statistics* 17(4):315–339.

Radu Ionescu, Marius Popescu, and Aoife Cahill. 2014. Can characters reveal your native language? a language-independent approach to native language identification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Scott Jarvis, 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.

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

McCallum Jonas. 2013. autocorrect spelling. https://github.com/phatpiglet/autocorrect.

Moshe Koppel, Jonathan Schler, and Kfir Zigdon. 2005. Automatically determining an anonymous author's native language. *Intelligence and Security Informatics* pages 41–76.

Shervin Malmasi and Mark Dras. 2017. Native language identification using stacked generalization abs/1703.06541. http://arxiv.org/abs/1703.06541.

Shervin Malmasi, Mark Dras, et al. 2014. Language transfer hypotheses with linear SVM weights. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1385–1390.

Shervin Malmasi, Keelan Evanini, Aoife Cahill, Joel Tetreault, Robert Pugh, Christopher Hamill, Diane Napolitano, and Yao Qian. 2017. A Report on the 2017 Native Language Identification Shared Task. In *Proceedings of the 12th Workshop on Building Educational Applications Using NLP*. Association for Computational Linguistics.

Shervin Malmasi, 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*. Association for Computational Linguistics, pages 124–133.

Sean Massung, Chase Geigle, and ChengXiang Zhai. 2016. MeTA: A Unified Toolkit for Text Retrieval and Analysis. In *Proceedings of ACL-2016 System Demonstrations*. Association for Computational Linguistics, pages 91–96.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space http://arxiv.org/abs/1301.3781.

Lourdes Ortega. 2014. *Understanding second language acquisition*. Routledge.

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

Ria Perkins. 2015. Native language identification (NLID) for forensic authorship analysis of weblogs. *New Threats and Countermeasures in Digital Crime and Cyber Terrorism* pages 213–234.

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

Björn Schuller, Stefan Steidl, Anton Batliner, Julia Hirschberg, Judee Burgoon, Alice Baird, Aaron Elkins, Yue Zhang, Eduardo Coutinho, and Keelan Evanini. 2016. The INTERSPEECH 2016 computational paralinguistics challenge: Deception, sincerity & native language. pages 2001–2005.

Joel Tetreault, Daniel Blanchard, and Aoife Cahill. 2013. A Report on the First Native Language Identification Shared Task. In *Proceedings of the Eighth Workshop on Building Educational Applications Using NLP*. Association for Computational Linguistics.

Joel Tetreault, 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 24th International Conference on Computational Linguistics*. Association for Computational Linguistics, pages 2585–2602.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*. Association for Computational Linguistics, pages 173–180.

Oren Tsur and Ari Rappoport. 2007. Using classifier features for studying the effect of native language on the choice of written second language words. In *Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition*. Association for Computational Linguistics, pages 9–16.

Mike Wallace. 2007. *Jawbone Java WordNet API*. http://mfwallace.googlepages.com/jawbone.

Xingyou Wang, Weijie Jiang, and Zhiyong Luo. 2016. Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In *Proceedings of the 26th International Conference on Computational Linguistics*. pages 2428–2437.

Robert Weide. 2005. The Carnegie Mellon pronouncing dictionary. http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

Jason Weston and Chris Watkins. 1998. Multi-class support vector machines. Technical report, Department of Computer Science, University of London.