Native Language Identification

Sophia Davis

LING 550 Final Project

sophia.davis3@mail.mcgill.ca

November 30, 2016

Native Language Identification

NLI, or guessing a speaker's mother tongue based on a sample of his written or spoken English, is already a fairly well-studied topic. Applications include:

- Tailored education: changing ESL instruction to correct errors made by speakers with different mother tongues.
- Linguistic knowledge: better understanding of the processes of transfer and language acquisition.
- Forensic linguistics: using NLI to uncover the identity of anonymous threats.

Previous Work in NLI

- Moshe Koppel 2005: NLI pioneer; focus on function words, character n-grams, spelling errors. Used linear Support Vector Machine to distinguish among 5 languages with $\sim$80% accuracy.
- NLI Shared Task 2013: 29 teams used various methods and models. Among methods: word and character n-grams, syntactic features. Best performing team able to identify correct language out of 11 $\sim$83% of the time.

The Data

Data were taken from the essay section of the Test of English as a Foreign Language, a standard benchmark of English proficiency.

- 12,100 essays of roughly 400 words each written by non-native English speakers of varying proficiencies
- Speakers' Native Languages include: Arabic, Chinese, French, German, Hindi, Italian, Japanese, Korean, Spanish, Telugu, and Turkish.
- Each group was represented equally in the training and test sets (1,100 essays from each group).

Minimum and Ideal Outcomes

- Any working model, however bad, would have to have over $1/11 = 9.09\%$ accuracy.
- ETS Native Language identification Challenge: teams' accuracy ranged from 30% to 83%. Most teams achieved accuracy of 75-80%.
- My minimum goal was to perform better than the worst team in the challenge (over 30%). My ideal outcome was the 83% achieved by the top performer.

Steps to a Working Model

- Consider which features will be the most predictive.
- Extract features from the training set.
- Train a model on those features, use it to predict native language of essays in validation (and eventually testing) set.

Features

Character bigrams

- To reduce featureset, only included a-z as well as common characters !?'., . Used in nearly all models in ETS open challenge.

Word Unigrams

- Only used words that occurred over five times across training set. Idea from existing work, including ETS challenge.

My idea: Levenstein deltas (Not currently implemented.)

- Systematically represent misspellings. More telling than individual misspellings.
- e.g. for the misspelling "ingineer" for "engineer" levenstein delta would be "-i+e" to say "replaced i for e"

Implementation

Scikit.learn: Implementation of Machine Learning techniques

- Python
- “Plug and chug” formulas make training easy. Hardest is prepossessing data, converting to array
- Used or Attempted: Linear SVC, Bagging Classifier, Random Forest, Decision Trees, SVM
- I won’t demonstrate because the time it takes to run is longer than my presentation, and it isn’t very exciting.

Success Rate

- Character bigrams with linear SVC: roughly 50% cross validation on training data. Same model with word unigrams: 65%. With both features: also 65%.
- Same data with more complex SVM: fail, took far too long to run.
- Experimentation with bagging raised accuracy to 69%
- Random forest (like decision trees with correction for overfitting): roughly 55% accuracy
- Code available at https://github.com/Sophia-Davis/nli

Potential Avenues of Improvement

Avenues to improve model performance

- Levenstein deltas/ other features....n-grams, syntactic features
- Look into which languages in particular are being misidentified, research which mistakes those speakers in particular make in order to correct them.

Questions?