Natural Language Processing of the Modern Irish Language

Scott A. Lewis
Department of Computer Science
Saint Louis University
St. Louis, MO, 63101

Abstract

The Irish language can be characterised by its use of diacritic accent symbols above letters as in á, é, í, ó, and ú that mark the instance of a long vowel sound. These long vowel notations are referred to as *síntí fada* in Irish. Aside from signifying a long sounding vowel sound, these diacritics alter the meaning and contextual usage of a word when switched in with their short vowel, ASCII counterparts. Here, we test multiple machine learning models in order to predict the usage of a *síntí fada* containing word and its ASCII counterpart. We implement four different approaches to estimating the probability of a given word. These approaches include basic likelihood estimation, TFIDF, logistic regression, and support vector machine. We also evaluate the effects of using ranges of TFIDF ngrams as well as common Irish language stop words on the Log-Loss estimated from probability predictions.

Data

The data consist of 456,571 Irish sentences scraped from the web. Each sentence contains at least one use of the 50 target words for prediction within the present study, and each of these 50 words can be found a minimum of 1,000 times within the corpus. Although each word is guaranteed to appear at least 1,000 times during training, the distribution of training sentences representing each ASCII/Diacritic word pair is uneven, due to some of the words being more or less common than others. The frequencies and distributions of training sentences for each of the twenty-five word pairs are represented in Table 1.

Word	Sentence Count
a	24200
ais	24200
aisti	2890
ait	24200
ar	24200
arsa	24200
ban	7269
cead	24200
chas	24200
chuig	24200

dar	24200
do	24200
gaire	6068
i	24200
inar	24200
leacht	3396
leas	24200
mo	24200
na	24200
os	24200
re	12105
scor	11497
te	16563
teann	5049
thoir	4534
TOTAL	456571

Table 1: Order and distribution of sentences containing target words.

Methods: Data Processing

Loading the Data:

Iterating through the training corpus, sentences for each of the twenty-five word pairs were collected and added to a nested dictionary at a level named after the word pair's ASCII value. Each sentence was also assigned a training label of 1 or 0 depending on if the sentence contained the ASCII or diacritic value of the pair, respectively. Sentences containing both the use of the ASCII value and the diacritic value were copied twice, one copy with the ASCII label and one copy with the diacritic label.

Sentences in the test corpus contained the regular expression {%s|%s}, where the fist string in the brackets is the ASCII value of one of twenty-five word pairs and the second is the corresponding diacritic value. In order to mark the location of the word the model is intended to predict, the first bracket in the regular expression was replaced by the marker symbol "漢" and the rest of the non-letter characters were stripped away. Sentences were then added to a nested dictionary that contained sub keys for the sentence, and the line number index where the sentence was pulled from in the test corpus, which would later be used for writing a submission file.

Data Cleaning:

Prior to vectorization, all sentences in training dictionary and test data were stripped of digits, punctuation, new line expressions, and other special characters. All words in sentences were made lowercase.

Methods: Models

Likelihood Estimation:

As a baseline, the first algorithm we implemented was a global likelihood estimation calculated by Equation I.

Equation I

$$p = \frac{ASCII_{count}}{(ASCII_{count} + DIAC_{count})}$$

Although naïve, this estimation achieved a log loss score of 0.45815, just 0.001 below the unigram-prior baseline.

Model	Log-Loss
Likelihood Estimate	0.45815

Table 2: Loss score obtained by a baseline likelihood estimation model

TFIDF:

For the present study, we used the Term Frequency Inverse Document Frequency (TFIDF) to vectorise sentences before fitting them to a true classifier. TFIDF is a method used to reflect how important a word is to a document in a corpus. In its most basic sense, TFIDF is a score that may be used to approximate the relative importance of a word and is calculated from the frequency of a word in a sentence normalised by the number of sentences containing that same word in the corpus. For this study we use the TfidfVectorizer() function from Scikit Learn, which gives the user the ability to regulate the minimum document frequency, stop words that should be ignored from scoring, as well as a range of ngrams to be included in the vectorisation. For the remaining classification algorithms, we used TFIDF-fit vectorisations of training sentences as input to calculate probabilities.

Logistic Regression:

The next model we implemented was logistic regression. We implemented this model with a max iteration parameter of 1000, and with the binary parameter set to true. We also evaluated the model after filtering out words from a list of common Irish stop words,

low frequency words, using TFIDF bigram (ngram = (2,2)), and TFIDF unigram+bigram (ngram=(1,2)).

Model	Log-Loss
Logistic Regression w/ common Irish stop words	0.31437
Logistic Regression w/ filtering	0.22441
Logistic Regression, ngram = (1,2)	0.23946
Logistic Regression, ngram = (2,2)	0.29733

Table 3: Log-Loss scores obtained by different word frequency filtering & TFIDF vectorisation parameters.

The best results were obtained from filtering out words that were found in fewer than 10 documents, while expanding the TFIDF ngram range only worsened the Log-Loss of this implementation.

Support Vector Machine:

The final model we evaluated was the SVC classifier. We implemented a probabilistic SVC classifier using a grid search of C range [0.01, 0.1, 1, 10] and gamma range ['auto', 0.01, 0.1, 1] to find the optimal values of the C and gamma hyper-parameters for the binary classification of each of the 25 ASCII/Diacritic word pairs.

Model	Log-Loss
SVC *w/ opt*	1.08796

Table 4: Log-Loss score obtained from optimised SVC classifier.

Surprisingly, this SVC implementation performed far worse than the baseline or of any other models evaluated in this study.

Results & Discussion

In this study we applied three different probabilistic classification algorithms to the problem of modelling the use of diacritics in the Irish language. We evaluated these algorithms at using some domain knowledge of common stop words in the Irish language, as well as using different ngram ranges in the vectorisation step. Surprisingly, we found that our optimised SVC model underperformed compared to a simple logistic regression algorithm and that adding ranges to the ngrams at the vectorisation step did not improve Log-Loss scores for the logistic regression model.

Conclusion & Future Work

Although we have shown that a simple logistic regression model can perform reasonably well when applied to this problem, we have not identified the global optima. A Long-Short Term Memory (LSTM) model may prove more effective in future studies, if given a large amount of training data.