Knowledge of Language Origin Improves Pronunciation Accuracy of Proper Names

Ariadna Font Llitjos and Alan W Black

Language Technologies Institute
Carnegie Mellon University
{aria,awb}@cs.cmu.edu

Abstract

As it is impossible to have a lexicon with complete coverage, and a high proportion of unknown words are proper names, this paper addresses the issue of automatically finding pronunciations of unseen proper names in US English. Proper names, especially in the US, may come from a large range of ethnic backgrounds. We present a model and results showing that including ethnic origin of words in a statistical model can improve pronunciation results.

We used a lexicon of 56,000 proper names from CMUDICT. We also gathered data (text and proper names) from 26 languages to built statistical models that provide an estimate of word origin.

Tests against held out data showed a 7.6% absolute improvement from a baseline of 54.8% when language based features were added to our CART-based model. As there are potentially multiple correct pronunciations, we synthesized a random sample of names that did not match the “correct” answer in our test set. Human listeners showed a 17% preference for the model with language features compared to the baseline.

1. Introduction

Our hypothesis is that, in the same way people adapt their pronunciation according to where they think a proper name comes from, if we add that knowledge to a statistical model of pronunciation, we should get higher pronunciation accuracy.

It is possible that, since many of the languages these names originate from have a more standardized pronunciation than English (e.g. Polish, Italian, Japanese), knowing the origin of an unknown word may allow more specific rules to be applied ([1] and [3]). In some cases, we even expect our system to outperform native American English speakers, since the average American English speaker does not have much knowledge about languages other than English. One such case is Chinese names, since few native English speakers know how to pronounce them, but there are very concrete English rules for pronouncing such names. If we added this information to our system, it would pronounce those names in the Americanized, educated way, achieving higher pronunciation accuracy than average American speakers.

What we are trying to model is the educated pronunciation of foreign proper names in American English, and not the original pronunciation of foreign words (which might be as puzzling to the American ear as a wrong pronunciation). For example, if we consider the proper name ‘Van Gogh’, what we want our system to output is not /F AE1 N G O K/ or /F AE1 N G O G/, which some people may claim is the correct way of pronouncing it, but rather the American pronunciation of it: /V AE1 N . G OW1/.

As Kenneth Church points out in [2], people in the US tend to adopt a *pseudo-foreign accent* (different than both the accent in the original language and the American English accent), which people are able to recognize as the (mis)pronunciation of foreign names. So what we are interested in approximating as much as possible with our model is the educated, Americanized pronunciation of foreign proper names.

For this reason we restricted ourselves to the set of American English phones as defined in CMUDICT [3], but we allowed more letter to phone alignments than the one used for the whole CMUDICT, which resulted in almost the double of phone combinations.

In general, and as also pointed out in [1], the reason we want to build an automatic pronunciation-predicting model for TTS systems is that, even if dictionaries are the most reliable methods, it is not possible to guarantee that a large lexicon will contain all the words found in a text. Thus, a fallback position is required that predicts the pronunciation form the orthographic form.

Previous experiments on large lists of words and pronunciations have shown that when a lexicon has more foreign words than another (CMUDICT vs. OALD in [1]), this has quite an impact on speech synthesis accuracy. Such experiments report a difference of 16.76% on word accuracy which can be attributed to proper names, since they are harder to predict without any higher level of information. Therefore, there is clearly room for improvement in this domain.

One could argue that, in real text, foreign words account for a small percentage of all the words, and so improvement in this area would have no significant impact on the overall accuracy of the system. However, we argue that, even if the amount of foreign names were relatively small, getting them right would substantially improve perceived synthesis quality. Moreover, some real applications deal mostly with proper names, and therefore it is worthwhile exploring this hypothesis (e.g. directory assistance, customer records, appointments etc.).

For practical purposes, we consider the knowledge of where a proper name comes from to be equivalent to what language it belongs, not in the sense of strict etymologic origin but rather in the sense of a proper name commonly used in a particular language. We know this not to be true, and determining "what language a proper name belongs to" has many more intricacies than it might be apparent. Jewish names are a good example of this, since they are common in many countries and therefore many languages. Likewise, *Michael* is a common proper name in at least English, German and French. For this reason, we implemented a language identifier that, given a word, it will give us the probability of that word belonging to all the languages we have letter language models (LLM) for, rather than just giving the *maximum a posteriori* (MAP) solution.

2. Previous related work

Black et al. [1] used the LTS rules technique on four lexicons, the Oxford Advanced Learners Dictionary of Contemporary English (OALD) (British English), CMUDICT version 0.4 (US English), BRULEX (French), and the German Celex Lexicon (DE-CELEX). Chotimongkol and Black [4] tried a variation of the same model with augmentation for solving letter to phone alignment problems in Thai (see table 1 for results).

The work that most closely relates the one presented here is Tomokiyo's unpublished project report ([5]). Tomokiyo tried using C4.5 and Maximum Entropy approach on the CMUDICT and OALD, which got lower results than the CART tree technique used in [1] and in this paper. After an error analysis, Tomokiyo observes that the existence of foreign words is a major cause of errors, and so he proceeds to focus only on the foreign word problem. He tried two different ways of adding ethnic origin or language class information: a character-based n-gram model plus k-means clustering, and adding n-letter suffixes as additional features for the ME model.

For C4.5 and unpruned trees, the letter accuracy went up 0.1% with the clustering approach and 0.3% by adding suffixes as additional features, whereas for the ME model, adding 3-letter suffixes improved letter accuracy in 1.4% (up to 89.9%).

Even if he did not get encouraging results, his work is most valuable for our research since he used the same data, and he explored the same problem using other techniques, obtaining results that are directly comparable to ours.

3. Baseline CART

For our proper name baseline, we used a list of proper names that came from Bell Labs' directory listings (from at least 20 years ago), which is supposed to contain the 50K most frequent surnames and 6k names in the US, and their pronunciation as it appears in the CMUDICT with stress (version 0.4).

We held out every tenth word in the 56,000 name list for testing and used the remaining 90% as training data.

Based on the techniques described in [6], we used decision trees to predict phones based on letters and their context. In English, letters map to epsilon, a phone or occasionally two phones. The following three mappings illustrate this:

(a) Monongahela	m ah n oa1 ng g ah hh ey1 l ax
(b) Pittsburgh	p ih1 t ε s b ε er g ε
(c) exchange	ih k-s ch ε ey1 n jh ε

where (a) has the same number of letters as phonemes, (b) has several letters map to epsilon ε (i.e. not pronounced), and (c) has the letter 'x' realized by two phones /k-s/. Stress is marked by adding a 1 after the stressed syllable.

First, we added additional multi-phones that appeared aligned to each letter and were not in the set of alignments allowed. In this case, instead of the 45-50 US phones, we ended up with 101 phone combinations. Most of the extra phones were vowel variations, and some were adding a vowel between two consonants. For example, for the letter 'e', the standard set of phones would be: */ih ih1 iy iy1 er er1 ax ah1 eh eh1 ey ey1 uw uw1 ay ay1 ow ow1 y-uw y-uw1 oy oy1 aa aa1/,* but for this particular data, we also added */ax ey-ih ao ae/,* since they appeared as the pronunciation of 'e' in some of the foreign names (e.g. *erle -> ao r ax l*); and for the letter 'c', the standard set of phones would be: */k ch s sh t-s,* and the ones we needed to add for this domain were */s-iy k-ax g/* (e.g. *cdebaca -> s-iy d ih b aa k ax,* and *csaszar -> k ax s aa sh er*).

Before training the decision trees, we aligned each letter to its corresponding zero or more phones. We found all possible alignments given the set of allowable letter/phone group mappings and calculated the likelihood of each. Then, we used that information to score all possible alignments and select the most likely as the actual alignment.

Then, from these alignments, the CART technique was used to train a decision tree for each letter. Up to three letters preceding and following the considered letter are used as the context for predicting phones.

As expected, the results for the CART trained on proper names (PN-base) are 3.72% lower than the ones for the whole CMUDICT, since it is harder to automatically find LTS rules for proper names than it is for other words:

Table 1: Previous results plus baseline for proper names

Lexicons	Letters	Words
OALD	95.80%	74.56%
CMUDICT	91.99%	57.80%
BRULEX	99.00%	93.03%
DECELEX	98.79%	89.38%
Thai	95.60%	68.76%.
PN-base	89.02%	54.08%

In this context, letter accuracy is defined as the number of letters that are correctly converted to epsilon, a phone, or multi-phone. Word accuracy is defined as the number of words where all phones (once re-split) match exactly all the phones in the entry of the test data, i.e. a held out part of the CMU dictionary of pronunciation.

This method of calculating the error takes into account epsilon alignments, so even if the generated string is correct, if the epsilon is aligned differently than in the test data entry, an error is counted, and the same is true for stress. For example, if we are taking stress into account and the system outputs the correct pronunciation but does not get the stress right, it will be counted as an error. In the worse case, it can happen that the system outputs the correct pronunciation, but since it is incorrectly encoded in the CMUDICT, it is counted as an error.

There is always some variability in the way different transcribers transcribe different words, and one mapping is not necessarily more correct than another. This is particularly true for foreign words, which have been transcribed by different transcribers, or may have common segments pronounced differently for historical or regional reasons. Such inconsistencies place an upper bound on the amount of improvement possible using the CMUDICT.

In spite of the noise, we assume the data is basically correct and that our statistical training methods will generate pronunciations that are at least reasonable approximations given the data. This assumption is supported by the results given below, where we show that even when the pronunciation predicted is wrong compared to held out data, it may still be acceptable to a human listener.

4. Letter to Language Models and Language Identifier

4.1. Data Collection

We tried to hand label 10% of the training set (516 names) to have a better idea what the language distribution we needed to cover with the language models was, but we found that we could only tag 43% of that data confidently[1]. The distribution for that part of the data was the following[2]:

> German (96), English (73), Italian (43), French (40), Polish (23), Swedish (13), Scottish (12), Spanish (10), Hebrew (7), Dutch (6), Catalan (6), Danish (5), Japanese (5), Irish (5), Russian (5), Chinese (2) and Portuguese (2)

We then proceeded to collect data for these languages (except for Irish and Scottish) plus a few more to build the letter to language models to build a language identifier and add the relevant features to the CART.

The data we eventually used for the LLMs was 18 corpora from the *European Corpus Initiative Multilingual Corpus I* (after converting all special characters to their ascii equivalent):

> English, French, German, Spanish, Croatian, Czech, Danish, Dutch, Estonian, Hebrew, Italian, Malaysian, Norwegian, Portuguese, Serbian, Slovenian, Swedish and Turkish.

with sizes ranging from 255 thousand to 11 million words, and 8 proper names corpora we built by crawling the web automatically, using the *Corpusbuilder* [7], as well as manually:

> Catalan, Chinese, Japanese, Korean, Polish, Thai, Tamil and other Indian languages (except for Tamil)

with sizes ranging 500 to 6198 names.

4.2. LLMs and Language Identifier

Our Language Identifier is a variation of the algorithm presented in [8] that only uses trigrams, since this was shown to be effective for a similar classifying task in [9].

Every language LLM consists of a table of all the possible trigrams and their relative frequencies estimated from the corpus for that language (sliding trigrams) and taking into consideration wether it occurs at the beginning or at the end of the word. We used Laplace smoothing, which only made a significant difference for the proper name corpora, since there is not enough data to reliably estimate all the trigrams for those languages.

The language identifier applies a LLM on the fly for the input word (or document) and, for every trigram in the input, it calculates the probability of it belonging to all the languages by multiplying them by the relative frequencies for those trigrams in each one of the languages (LLMs).

Rather than using this language identification in a direct way by building trees explicitly for each language, which due to sparseness of data would not be ideal, we use the results from the language identification process as features within the CART build process, thus allowing those features to affect the tree building only when their information is relevant.

Given a name, the language identifier gives us the probabilities of that word belonging to all 26 languages. To build the CART we decided to only add the following features to each name: 1st-language, higher-probability, 2nd-language, 2nd-higher-probability and the difference between the two higher probabilities. An example of input to the CART is:

(zysk ((best-lang slovenian.train) (higher-prob 0.18471)
(2nd-best-lang czech.train) (2nd-higher-prob 0.18428)
(prob-difference 0.00043))   )

5. Results

We built the CART (PN-lang) using 5 as stop value, which had been proven optimal for the CMUDICT in previous experiments [1]. However, since the parameter space was

[1] The data is ascii only, and all the accent marks and special characters (which are excellent clues for language identification) had been removed. Other reasons hand labeling was hard was that many of the names belong to many different languages, and that nobody had heard of many of the names in the list.

[2] Because of the way we hand labeled the data, it might have resulted into slightly boosting the languages we are more familiar with and the languages for which we had native speakers to look at the list first.

richer (the tree had more features to split itself into), we suspected there was a data fragmentation problem, and there wasn't actually enough data on the leaves to have reliable estimates, so we also built CARTs using a stop value of 8. The word accuracy for all the CARTs were the following:

Table 2: Proper name experiments results

Lexicons	Letters	Words
PN-base-5	89.02%	**54.08%**
PN-lang-5	91.23%	**61.72%**
PN-base-8	90.29%	52.88%
PN-lang-8	90.63%	59.77%

Which represents a 7.64% increment in word accuracy over the proper name baseline. Note that both PN-lang-5 and PN-lan-8 have higher word accuracy than the model trained on the whole CMUDICT (see table1).

6. User studies

From the names that both PN-base-8 and PN-lang-8 got "wrong" (did not exactly match the CMUDICT pronunciation in the test set), we selected the ones for which the two models assigned a different pronunciation. From those, we picked 20 names at random and synthesized them to run perceived accuracy user studies.

In the user studies, we asked users to score the two different pronunciations for each one of the 20 names from 1 (very inaccurate – not understandable) to 5 (very accurate – good approximation of educated American English pronunciation for that name). Overall, the perceived accuracy of the probability model was 17% higher (PN-lang-8: 46%, PN-base-8: 29%, no preference: 25%).

These results are based on 23 native American English speakers with a high level education (from undergraduates to professors), 15 of which had some knowledge of languages other than English, and all of which had been exposed to synthesized speech before.

The results for non-native speakers also showed the same increase in perceived accuracy for the model that incorporates language probability information.

7. Conclusions

Language probability information definitely improves pronunciation accuracy of proper names. However, there are still many experiments that need to be done to find out what is the upper bound in accuracy when following this approach.

Ideally, we should have trained our LLMs on just names, instead of text corpora, since that is the distribution of our training data. However, some experiments where we had LLM trained on both text and just proper names for German, French and Spanish have shown that the probability of the two LLM were very close, and it never happened that the LLM trained on text performed worse than the LLM trained on proper names.

Another experiment we need to try is adding prior probabilities. For each language, we would have a prior probability that would tell us how likely it is to find a name in that language, independently of the name. If our model were trained from newswires data instead of a name directory, it would be relatively easy to determine such priors. For example, if we had "*Yesterday in **Barcelona**, the mayor Joan Clos inaugurated the Forum of Cultures…*", then our prior *P(Catalan)* would go up to 0.8 say, and *P(Spanish)* would go up to 0.15, whereas the prior probabilities for all other languages would be very close to 0.

With our training data, however, it is very hard to determine such priors if we do not know exactly from which distribution the names come from. If we had all the names for CMU students, faculty and staff, say, then we could look up the country of origin statistics and set our priors accordingly.

Human language identification in isolation (without having any contextual information nor a prior) is very hard. Humans can confidently tag less than 50%. Therefore, a system that mimics human behavior when pronouncing a foreign proper name but that uses an automatic language identifier is almost certainly going to perform better, and it will surely be more consistent.

8. Acknowledgement

This work was supported in part by a "La Caixa" Fellowship 2000-2001.

9. References

[1] Black, A., Lenzo, K. and Pagel, V. *Issues in Building General Letter to Sound Rules*. 3rd ESCA Speech Synthesis Workshop, pp. 77-80, Jenolan Caves, Australia, 1998

[2] Church, K. (2000). *Stress Assignment in Letter to Sound rules for Speech Synthesis* (Technical Memoradnum). AT&T Labs –Research. November 27, 2000.

[3] CMUDICT. Carnegie Mellon Pronunciation Dictionary. 1998. http://www.speech.cs.cmu.edu/cgibin/cmudict

[4] Chotimongkol, A. and Black, A. *Statistically trained orthographic to sound models for Thai*. Beijing October 2000.

[5] Tomokiyo, T. *Applying Maximum Entropy to English Grapheme-to-Phoneme Conversation*. LTI, CMU. Project for 11-744, CMU unpublished. May 9, 2000.

[6] Black, A. Festival Manual: *Building letter to sound rules*. http://www.speech.cs.cmu.edu/festival/manual-1.4.1/festival_13.html#SEC44

[7] Ghani R., Jones R. and Mladenic D. *Building Minority Language Corpora by Learning to Generate Web Search Queries*. Technical Report CMU-CALD-01-100, 2001. http://www.cs.cmu.edu/~TextLearning/corpusbuilder/

[8] Canvar, W.B., and Trenkle J.M. *N-Gram-Based Text Categorization*. In Proceedings of 3rd Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, NV. April 13, 1994.

[9] Sproat, R. et al. (1999). *Normalization of Non-standard Words WS '99 Final Report*. John Hopkins University Workshop 99, CLSP, Pronunciation Group.