Pair Language Models for Deriving Alternative Pronunciations and Spellings from Pronunciation Dictionaries

Russell Beckley
Oregon Health and Science University
beckleyr@ohsu.com

Brian Roark
Google Inc.
roarkbr@gmail.com

Abstract

Pronunciation dictionaries provide a readily available parallel corpus for learning to transduce between character strings and phoneme strings or vice versa. Translation models can be used to derive character-level paraphrases on either side of this transduction, allowing for the automatic derivation of alternative pronunciations or spellings. We examine finite-state and SMT-based methods for these related tasks, and demonstrate that the tasks have different characteristics – finding alternative spellings is harder than alternative pronunciations and benefits from round-trip algorithms when the other does not. We also show that we can increase accuracy by modeling syllable stress.

1 Introduction

Robust processing of speech and language requires dealing with variation in language production, either in terms of pronunciation in the spoken domain or spelling in the written domain. Predicting the intended words of an acoustic or textual sequence is an important recognition task, often required for downstream processing such as spoken language understanding or knowledge extraction. Informal text genres, such as those found in social media, share some characteristics with speech; in fact such text is often informed by pronunciation variation. For example, consider the following tweet:

> He aint gotta question my loyalty, cuz he knw wen sh!t get real. Ill be right here!

where several tokens (e.g. "cuz", "wen") represent spelling alternations related to pronunciation. Work in text normalization and spelling correction – e.g., Toutanova and Moore (2002); Li and Liu (2012) – has included pronunciation information to improve recognition of the intended word, via grapheme to phoneme (g2p) conversion modeling derived from pronunciation dictionaries.

Pronunciation dictionaries provide natural parallel corpora, with strings of characters paired to strings of phones. Thus, standard lexicons have been used in recent years with machine translation systems such as Moses (Koehn et al., 2007), to train g2p systems (Laurent et al., 2009; Gerosa and Federico, 2009). Further, other algorithms using such dictionaries also use translation phrase tables, but not for translation tasks. For example, data-driven paraphrasing methods (Bannard and Callison-Burch, 2005) use translation phrase-tables as a "pivot" to learn sets of phrases which translated to the same target phrase. In a similar manner, with a pronunciation dictionary instead of a phrse-table, pivoting can be used to learn alternative pronunciations (Karanasou and Lamel, 2010), i.e., direct phoneme-to-phoneme (p2p) "translation" systems that yield alternative pronunciations. Alternatively, round-trip translation could be used, e.g., to map from letter strings to phone strings in one step, then from the resulting phone strings to letter strings in a second step, as the means to find alternative spellings (Li and Liu, 2012).

In this study, we explore dictionary-derived models to find either alternative pronunciations or alternative spellings, using either direct (p2p or g2g) or round-trip algorithms (p2g2p or g2p2g). We compare methods based on weighted finite-state transducers (WFST) with phrase-based models trained with Moses. Our main interest is to evaluate Karanasou and Lamel (2010) methods – shown to be useful for deriving alternative pronunciations – for deriving alternative spellings, and thus to determine the relative difficulty of these two tasks. We also examine when, if ever, round-trip processing yields benefits over direct transduction. Our results indicate that real alternative pronunciations are substantially easier to find than real alternative spellings, particularly when pronunciation features such as syllable stress are available. Second, round trip translation yields no gain (and some loss) over direct transduction for finding alternative pronunciations, yet yields some modest gains for finding alternative spellings. Further, WFST methods perform as well as or better than Moses trained models. Finally, combining the methods yields further gains, indicating that the models are learning complementary sets of patterns.

The primary contribution of this work is to introduce a competitive method of building and using pair language model WFSTs for generating alternative spellings and pronunciations which reflect real-world variability. This could improve results for downstream processes, e.g., epidemiological studies (Chew and Eysenbach, 2010) or sentiment analysis (Barbosa and Feng, 2010) derived from social media text. Further, we present a controlled comparison between the two tasks, and demonstrate that they differ in terms of task difficulty

2 Related work

Text normalization has been a major focus in text-to-speech (TTS) research for many years. Notably, Sproat et al. (2001) deemed it a problem in itself, rather than *ad hoc* preparatory work, and defined many of the issues involved, as well as offering a variety of initial solutions. Similar approaches apply to automatic spelling correction, where Toutanova and Moore (2002) extended the noisy channel spelling correction method of Brill and Moore (2000), by modeling pronunciation alternations to infer from misspellings to correct spellings. Similarly, Li and Liu (2012) extended the character-based translation approach to text normalization of Pennell and Liu (2011), by adding an additional round-trip translation to-and-from pronunciations. Karanasou and Lamel (2010) used Moses to generate alternative pronunciations from an English dictionary, using both direct and round-trip methods. They validated their systems on a set of words with multiple pronunciations, measuring the degree to which alternative pronunciations are generated from one of the given pronunciations. Our task and method of evaluation is similar to theirs, though we also look at alternative spellings.

3 Methods

To generate alternative spellings and pronunciations, we built phrase-based translation and finite-state transduction models from a parallel corpus. When pronunciations were part of the model – i.e., not direct grapheme-to-grapheme – we included conditions with and without vowel stress.

3.1 Corpus

Our training corpus is the CMU Pronouncing Dictionary[1], which contains nearly 130k entries. From this corpus, we identified homophone sets, i.e., sets of multiple spellings sharing the same pronunciation, such as "colonel" and "kernel". We found 9,977 such sets, and randomly selected 1000 for testing; the rest we used for training. Each set had, on average, 2.46 members. We also identified homograph sets, i.e., sets of multiple pronunciations all spelled the same, such as potato (/potato/ and /pәteto/). We found 8,216 such homograph sets, and randomly selected 1000 for testing; the rest we used for training. These sets averaged 2.13 members.

We construct seven parallel training corpora from the lexicon, each disjoint from its relevant test set. For round-trip models, the parallel corpus is each grapheme string in the lexicon aligned with its phoneme string, if neither the grapheme string nor phoneme string appear in the test set. There are four such corpora, corresponding to these options: stress or no stress, and g2p2g or p2g2p. The g2p2g and p2g2g conditions require different corpora because they are differently partitioned for testing. For direct grapheme-to-grapheme training sets, non-homophone words are self-aligned; for homophones, from each homophone set, each possible pair of spellings are aligned. For example, for a pronunciation with four spellings—a, b, c, and d—there would be six alignments: a:b, a:c, a:d, b:c, b:d, c:d. Similarly for direct phoneme-to-phoneme training sets, non-homograph words are self-aligned; words from the training homograph sets are pairwise aligned in all pairings. There are two direct p2p corpora: with and without stress.

3.2 Phrase-based translation models

As a baseline system, we used the Moses statistical machine translation package (Koehn et al., 2007) to build grapheme-based and phoneme-based translation systems, using a bigram language model.[2] These are trained on the parallel corpus resulting from the homophone or homograph sets detailed in

[1] http://www.speech.cs.cmu.edu/cgi-bin/cmudict

[2] Higher order language models yielded no improvements.

the previous section for the direct methods. For this paper, we did not perform round-trip translation with Moses, rather present it as a baseline for the direct approach.

3.3 Pair language models

Our weighted finite-state transducer approach is based on pair language models (Bisani and Ney, 2008; Deligne and Bimbot, 1997; Ghoshal et al., 2009), or, more recently, (Sagae et al., 2012).) The basic idea in a pair LM is to align strings, then train a language model over sequences whose symbols are the input:output pairs of the alignment. This language model can then be converted to transducers. For a g2g example, homophones "their" and "there" are aligned via the standard Levenshtein edit distance algorithm as "t:t h:h e:e i:ϵ r:r ϵ:e". A trigram model over these x:y strings would use standard n-gram modeling to estimate, for example, P(ϵ:e | i:ϵ r:r); i.e., the probability of a silent "r" in a given context.

Building the pair language model transducers requires two phases. In the first phase we create new corpora by aligning the elements of the parallel corpora outlined above. In the second phase we use these corpora of string alignments to build a pair language model.

3.3.1 Alignment and Corpora Building

We use extensions to the Levenshtein edit distance algorithm to align g2g, p2p and g2p strings, with substitution matrices created to provide useful alignments (Wagner and Fischer, 1974). As in Brill and Moore (2000), we allow for certain multi-symbol strings to be substituted with a single cost, e.g., substituting 'th' with /θ/ in g2p alignment. For g2g alignment, our substitution cost is 0 for identity and 2 for a few pairs of commonly interchangeable graphemes, such as 'c' and 'k'. Other substitutions are not permitted, and delete and insertion have cost 10. For p2p alignment there are two conditions, with and without stress. Without vowel stress, no substitutions other than identity are allowed; with vowel stress, substitution cost is 2.5 for the same vowel with differing stress; and 5.0 if substituting a vowel with another vowel. Other substitutions are not permitted, and, again, delete and insertion have cost 10.

For training round-trip models, we have to perform g2p and p2g alignment, with differing alphabets on the input and output of the alignment. We begin with a basic substitution table that allows graphemes and their most likely phonemes to align. We then re-estimate the substitution costs based on relative frequency estimation (-logP), and also aggregate sequences of consecutively deleted graphemes so that they collectively map to a single phoneme. For example, given the alignment 'o:/a/ u:/ϵ/ g:/ϵ/ h:/ϵ/ t:/t/', ('ought', /at/), we make a new rule: ough:/a/, and give it a cost based on its relative frequency. Grapheme strings that appear sufficiently often with a given phoneme will thus accumulate sufficient probability mass to compete.

Each alignment produced as described above is a string in a training corpus for creating a pair language model. As such, each alignment pair (e.g. a:/ə/) is a token.

3.3.2 From Corpora to WFSTs

We use the open source OpenGrm NGram library (Roark et al., 2012) to build 5-gram language models from the strings of input:output pairs. These langauge models are encoded as weighted finite-state acceptors in the OpenFst format (Allauzen et al., 2007). We shrink the models with the `ngramshrink` command, using the relative entropy method (Stolcke, 1998), with the "theta" threshold set at $1.0e-6$. These finite state acceptors are then converted into transducers by modifying the arcs: split the labels of each arc, x:y, making x the input label for that arc, and y the output label. Thus traversing such an arc will consume an x a return a y. Such pair language models we use for all WFST methods discussed here.

3.4 Producing k-best output

Each tested input string, spelling or pronunciation, is encoded as a cost-free linear chain WFST and composed with a pair language model transducer described in the previous section. The resulting lattice is converted to an acceptor by projecting onto its output labels, i.e., for each arc, the input label is set to the value of the output label. Epsilons are then removed and the result is determinized. The k-best paths are extracted using the shortest path algorithm in the OpenFst library.

For direct models (g2g and p2p), the k-best output from this first transduction is our result, ranked according the probability of each path. For round-trip methods (e.g. g2p2g), however, we do a second transduction in the other direction. For example, for g2p2g, the first transduction would have transduced from a spelling to a set of candidate pronunciations; the second transduction will transduce from pronunciations to spellings. For this second transduction, we take each string s from the k-best list from the first transduction, and process them as we did in the first transduction, now using the inverse transducer. So, for each s in the first k-best list, we now have a k-best list from the second transduction. Thus, for the original input string, we have up to k^2 alternatives. Finally, we score each alternative by combining their scores from both transductions.

Let $\bar{p}$ represent a phoneme string, and $\bar{g}$ a grapheme string. If we perform a transduction from $\bar{p}$ to $\bar{g}$, the weights from the transducer provide the (negative log) joint probability $\mathrm{P}(\bar{p}, \bar{g})$. By performing a soft-max normalization over the k-best list output, we obtain the (negative log) conditional probability $\mathrm{P}(\bar{g} \mid \bar{p})$. For round-trip methods, we take the product of the conditional probability in each direction, and marginalize out the intermediate grapheme sequence, i.e.,

$$\mathrm{P}(\bar{p}_2 \mid \bar{p}_1) \;=\; \sum_{\bar{g}} \mathrm{P}(\bar{p}_2 \mid \bar{g})\, \mathrm{P}(\bar{g} \mid \bar{p}_1).$$

4 Experimental results

For evaluation purposes, we reserved a set of 1000 test homophone sets and 1000 test homograph sets, as described in Section 3.1. From each set, we generate alternatives from the longest set member (ties broken alphabetically) and examine the resulting k-best list for presence of other members of the set. Note that the input string itself is not a target, and, before evaluation, is removed from the k-best list. Recall is the proportion of the k-best list returned by the system:

$$\text{Recall}(\{\text{k-best}\}) \;=\; \frac{\mid \{\text{k-best}\} \cap \{\text{gold-list}\} \mid}{\mid \{\text{gold-list}\} \mid}$$

.

Results for generating alternative pronunciations are listed in Table 1; those for generating alternative spellings are in Table 2. For alternative spellings, we also present results that combine the outputs of direct, round-trip (no stress) and Moses into a single list using a simple ranked voting scheme (simple Borda count).

A noteworthy result is the apparent usefulness of stress modeling for predicting pronunciation variation using WFSTs with the direct method; this is seen in the first two data columns of 1. This suggests that stress has an effect on phoneme alteration, something we discuss in more detail in Section 5.

Recall: Alternative Pronunciations						
k-best size	pair language model				Moses	
	Direct		Roundtrip		Direct	
	stress	none	stress	none	stress	none
1	0.43	0.54	0.38	0.37	0.44	0.46
3	0.77	0.71	0.59	0.58	0.60	0.62
5	0.82	0.77	0.66	0.66	0.64	0.65
10	0.86	0.80	0.73	0.76	0.68	0.69

Table 1: Recall for generating alternative pronunciations

However, while providing a large gain in the p2p condition, pronunciation modeling gives small or negative effects elsewhere. In the round trip methods, the effects of stress are lost: stress has little influence of how a particular phoneme is spelled. Thus, graphemes do not retain much stress information, hence any pass through the orthographic domain will shed it.

Recall is higher for alternative pronunciations than for alternative spellings. One reason for this is that spellings in our test set average eight letters, whereas the pronunciations average around five phonemes. Furthermore, the average Levenshtein distance between original spellings and their target alternatives, is 2.6, while for pronunciations, it is 2.2. Combining these factors, we see that, for spellings, more edit operations are required, and there are more symbols to which to apply them. Therefore, for spellings, there are more incorrect candidates.

The results also show gains resulting from the roundtrip method when applied to finding alternative spellings, but no such gains when roundtrip methods are applied to alternative pronunciations. Suppose, when seeking alternatives for some spelling, we alter grapheme g_1 to g_2. With a direct method, we must have instances of g_1 mapping to g_2 in the training set. The roundtrip method, however, is less constrained: there must exist some phoneme p_1 in the training set such that g_1 maps to p_1, and p_1 maps to g_2; thus, the set of possible alternations at testing are $\{g_1 \rightarrow p_1\} \times \{p_1 \rightarrow g_2\}$. This argument also applies to finding alternative pronunciations. Thus the roundtrip method offers more possible mappings. These extra possible mappings may be helpful or harmful, depending on how likely they are compared to the possible mappings they displace. Why are they helpful for alternative spellings, but not for al-

Recall: Alternative Spellings					
k-best size	pair language model			Moses	Comb.
	Direct	Roundtrip		Direct	Direct
	none	stress	none	none	none
1	0.19	0.19	0.19	0.20	0.30
3	0.36	0.38	0.37	0.39	0.52
5	0.45	0.49	0.48	0.48	0.60
10	0.55	0.63	0.62	0.60	0.69

Table 2: Recall for generating alternative spellings

ternative pronunciations? We discuss one possible explanation in Section 5.

Comparing Moses to the pair language model methods, Moses does slightly better for smaller n ($n = 1, 3$), and slightly worse for larger n ($n = 10$). Our only partial explanation for this is that Moses does well at weighing alternatives but, possibly, does not generate a large number of viable alternatives. System combination yields solid gains in finding alternative spellings, demonstrating that these different systems are coming up with diverse options.

Finally, we note that many of the false positive pronunciations given by the WFST system are plausibly correct although they are not included in the CMU dictionary. For example, for the spelling, *adequate*, the CMU dictionary provides two pronunciations: /ædəkwət/ and /ædəkwet/. Meanwhile, the p2p WFST system (with stress modeling) produces /ædəkwIt/. This suggests that we can learn from CMU dictionary to predict actual pronunciations that CMU dictionary does not itself list.

5 Discussion and Summary

The experimental results demonstrated the utility of stress modeling for generating alternative pronunciations, which we suggested was due to the impact of stress on phoneme alternation. To examine this more closely, we looked at each phoneme, stress class, (ph, s)—e.g. (/ə/, primary)—and determined how likely is an occurrence of (ph, s) to have an alternative phoneme in a homograph set. We found that primary and secondary stressed vowels had an alteration probability of 0.017, while non-stressed vowels had an alteration probability of 0.036. This difference should be picked up in the transition probabilities of our WFSTs, resulting in a preference for alterations of unstressed vowels. This is analogous to results found in (Greenberg et al., 2002) for spontaneous American English discourse. A further analysis of the system output might shed more light on relationships between stress and phoneme choice.

Why are round-trip methods useful for finding alternative spellings but not for finding alternative pronunciations? One possible explanation is that the variety of orthographic alternations is greater than that of pronunciation alternations. Thus, the training set for spelling may provide less relative coverage of the alternations in its test set than the training set for pronunciation provides for *its* test set. This is supported by the fact that pronunciation recall exceeds spelling recall. The roundtrip method allows for finding mappings not seen in training. These extra mappings might be no better for spelling than they are for pronunciation, but for spelling, the mappings they replace in the k-best list are worse, so they yield an improvement. For pronunciation, the mappings they replace in the k-best list are better, so they yield a loss. Further research is required to validate this explanation.

Ultimately, we would like to apply these methods to the normalization of social media text, especially to find alternative spellings based on alternative pronunciations. To apply such methods to, say, Twitter normalization requires a sizable corpus mapping canonical spellings to non-standard spellings. To assess domain portability, we applied a model built from the CMU dictionary to just over 100 alternative spellings observed in a small Twitter collection. Using the direct g2g method, we generated alternative spellings from the canonical spelling of each term, and measured the recall of the output, i.e., whether the observed alternatives were present in the k-best list. Recall was extremely low (less than 5%), suggesting that the type of orthographic alterations that are found in dictionary pronunciations are very different from the orthographic variations found on Twitter, and that those differences have a profound effect on our ability to recover alternatives.

In sum, we have presented a small study of the utility of pronunciation dictionaries for finding spelling and pronunciation alternatives, demonstrating key differences between these tasks.

Acknowledgments

This work was supported in part by NSF grant #BCS-1049308. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF.

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Proceedings of the Twelfth International Conference on Implementation and Application of Automata (CIAA 2007), Lecture Notes in Computer Science*, volume 4793, pages 11–23.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 597–604.

Luciano Barbosa and Junlan Feng. 2010. Robust sentiment detection on twitter from biased and noisy data. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 36–44.

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50(5):434 – 451.

Eric Brill and Robert C Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 286–293.

Cynthia Chew and Gunther Eysenbach. 2010. Pandemics in the age of twitter: content analysis of tweets during the 2009 h1n1 outbreak. *PloS one*, 5(11):e14118.

Sabine Deligne and Frdric Bimbot. 1997. Inference of variable-length linguistic and acoustic units by multigrams. *Speech Communication*, 23(3):223 – 241.

Matteo Gerosa and Marcello Federico. 2009. Coping with out-of-vocabulary words: open versus huge vocabulary asr. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 4313–4316.

A. Ghoshal, M. Jansche, S. Khudanpur, M. Riley, and M. Ulinski. 2009. Web-derived pronunciations. In *Proc. ICASSP*.

Steven Greenberg, Hannah Carvey, and Leah Hitchcock. 2002. The relation between stress accent and pronunciation variation in spontaneous american english discourse. In *In Proceedings of ISCA Workshop on Prosody in Speech Processing (Speech Prosody 2002), Aix-enProvence*.

Panagiota Karanasou and Lori Lamel. 2010. Comparing SMT methods for automatic generation of pronunciation variants. In *Advances in Natural Language Processing*, pages 167–178. Springer.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180.

Antoine Laurent, Paul Deléglise, Sylvain Meignier, and France Spécinov-Trélazé. 2009. Grapheme to phoneme conversion using an smt system. In *Proceedings of Interspeech*, pages 708–711.

Chen Li and Yang Liu. 2012. Normalization of text messages using character-and phone-based machine translation approaches. In *Proceedings of Interspeech*.

Deana Pennell and Yang Liu. 2011. A character-level machine translation approach for normalization of sms abbreviations. In *Proceedings of IJCNLP*, pages 974–982.

Brian Roark, Richard Sproat, Cyril Allauzen, Michael Riley, Jeffrey Sorensen, and Terry Tai. 2012. The OpenGrm open-source finite-state grammar software libraries. In *Proceedings of the ACL 2012 System Demonstrations*, pages 61–66.

Kenji Sagae, Maider Lehr, Emily Tucker Prud'hommeaux, Puyang Xu, Nathan Glenn, Damianos Karakos, Sanjeev Khudanpur, Brian Roark, Murat Saraclar, Izhak Shafran, Daniel M. Bikel, Chris Callison-Burch, Yuan Cao, Keith Hall, Eva Hasler, Philipp Koehn, Adam Lopez, Matt Post, and Darcey Riley. 2012. Hallucinated n-best lists for discriminative language modeling. In *ICASSP*, pages 5001–5004. IEEE.

Richard Sproat, Alan W Black, Stanley Chen, Shankar Kumar, Mari Ostendorf, and Christopher Richards. 2001. Normalization of non-standard words. *Computer Speech & Language*, 15(3):287–333.

Andreas Stolcke. 1998. Entropy-based pruning of backoff language models. In *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, pages 270–274.

Kristina Toutanova and Robert C Moore. 2002. Pronunciation modeling for improved spelling correction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 144–151.

Robert A Wagner and Michael J Fischer. 1974. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173.