Phonetic Normalization of Microtext

Richard Khoury
Department of Software Engineering
Lakehead University
Thunder Bay, Canada
Richard.Khoury@lakeheadu.ca

Abstract—Microtext normalization is the challenge of discovering the English words corresponding to the unusually-spelled words used in social-media messages and posts. In this paper, we propose a novel method for doing this by rendering both English and microtext words phonetically based on their spelling, and matching similar ones together. We present our algorithm to learn spelling-to-phonetic probabilities and to efficiently search the English language and match words together. Our results demonstrate that our system correctly handles many types of normalization problems.

Keywords—microtext; social media; normalization; phonetic; wiktionary

I. INTRODUCTION

The term “microtext” [1] describes the type of very short and informally-written digital text documents that have become omnipresent in today’s world, including notably as online posts on Facebook and Twitter. One challenge when dealing with microtext stems from their highly relaxed spelling rules and extreme irregularities in spelling. Indeed, a sampling of Twitter messages studied in [2][3] found over 4 million out-of-vocabulary (OOV) words, and new spelling variations are created constantly. The challenge of developing algorithms to automatically correct the OOV words found in microtexts and replace them with the correct in-vocabulary (IV) words is known as normalization. It was noted in [3] that OOV spelling variations seem to follow a small set of simple rules. The rules proposed in [3] are “abbreviation” (deleting letters from the word, for example spelling the word “together” as “tgthr”), “phonetic substitution” (substituting letters for other symbols that sound the same, such as “2” for “to” in “2gether”), “graphemic substitution” (substituting a letter for a symbol that looks the same, such as switching the letter “o” for the number “0” in “0gether”), “stylistic variation” (misspelling the word to make it look like one’s personal pronunciation, such as writing “todega” or “together”), and “letter repetition” (repeating some letter for emphasis, for example by typing “togetherrr”).

In this paper, we propose a new approach to dealing with normalization. Our underlying assumption is that microtext users recognize words not thanks to correct spelling but by sounding them out. Consequently, no matter how innovative microtext spellings get, the resulting OOV words must still be phonetically similar enough to the intended IV words in order for the readers to understand them. For example, the sentence “r u askin anyl b4 teh gamez 2nite” sounds like “are you asking anyone before the games tonight?” if read out loud, and is perfectly understood despite being composed exclusively of OOV words. We propose to tackle the challenge of microtext normalization by building an algorithm that can determine the most probable phonetic renderings of OOV words and match them to the most probable similar-sounding English words.

II. PHONETIC TREE

Our proposed algorithm will be trained to determine the probable pronunciation of English words based on their spelling. Then, when presented with a new OOV word, it will determine the most probable IV words with similar pronunciation. There are thus two challenges to consider: how to map spelling to probable pronunciation at the training stage, and how to efficiently search the English language for words with similar pronunciations to an OOV word at runtime.

The technique we used to train our system on how to map letters to sounds is to get a list of English words with their correct pronunciations to use as examples. For this list, we obtained a copy of the English Wiktionary [4] as an XML file. This makes it easy for a software to pick out individual articles and extract the word defined in each page and their pronunciations written with IPA phonetic symbols. For example, the word “about” has the standard pronunciation /ˈbaʊt/, two Canadian pronunciations /ˈbəut/ and /ˈbət/, and an Irish pronunciation /ˈbəut/. Our processing extracted 37,500 pronunciations of 30,368 different English words. There are 151 different groups of IPA symbols used to represent 230 different groups of letters in our training data.

The foundation of our normalization algorithm is a radix-tree-structured phonetic dictionary of the English language. Starting from the root, each word’s phonetic transcription is inserted symbol by symbol, with each symbol being a separate child in the tree. Every word can be read phonetically by following a path through the tree. A node which has the last phonetic symbol of a word will store the list of words that have the pronunciation represented by the path (including homophones). Moreover, the path may continue beyond that node, as some words can also be prefixes of longer words.

The tree is built by inputting each of the 37,500 phonetic transcriptions one by one, starting at the root and branching off as needed. While this is done, the algorithm also learns two sets of phonetic probabilities. The first is the probability of a (set of) phonetic symbol(s) given a (set of) letter(s). This is computed by counting the number of times a (set of) letter(s) is mapped to a (set of) phonetic symbol(s) in all its occurrences in the training data. The second is the probability of a phonetic symbol as a child of a given node in the tree. This is computed...
by counting the number of times (or words where) a phonetic symbol occurs as a child of another symbol on a specific path.

In order to allow our system to deal with the different types of OOV words and normalization challenges, we also defined certain additional data structures. First, the similar-sound list is a mapping of phonetic symbols to other symbols that sound similar to it. This will make it possible, for example, to learn that “a”, “e”, “A”, and “E” sound similar since they occur in the same position in “about”, and to recognize other words where one of these symbols replaces another. Next, to handle graphemic substitutions, we build a graphemic substitution list: 4 for A, 1 for I or L, 0 for O, 3 for E, and 7 for T. Finally, we included a word probability list in our system. The list chosen is the freely available list of Project Gutenberg [5].

To use this tree for phonetic microtext normalization, we implemented the uniform-cost search algorithm presented in Figure 1. It maintains a search list of edge nodes, each one tracking the list of letters in the OOV word that have not been rendered phonetically yet, the string of phonetic symbols rendered so far, the probability of the current phonetic rendering, and where in the tree that edge node is located. At each iteration, the search algorithm picks the edge node with the highest probability and finds all prefixes of letters that have phonetic symbol equivalents and mapping probabilities. Of these symbols, some will be valid children nodes with child probabilities. These may also be enhanced by adding similar-sounds from the list, to deal with stylistic variations. Graphemic substitutions and abbreviations are then handled by adding more letters or symbols. New edge nodes lose the prefix letters from the word but add the phonetic symbols, will have a new probability that will be the product of the previous node probability and the probabilities of the added phonetic symbol, and will be at an edge node corresponding to the child node.

**Initial edge node:** word = input OOV word; phonetic string = “” ; node probability = 1.0; current tree node = root node

**Search list:** { Initial edge node }

1. Get the node with the highest phonetic probability from the search list
2. If there are no letters in the word:
   a. Get the list of IV words at the current tree node
   b. Multiply the node probability by the IV word probability.
   c. Add IV word and probability to a list of normalized words
   d. If the normalized list achieves a termination condition, return it
   e. Otherwise, go to step 1
3. Get the known letters prefixes of the word
4. For each letter prefix from step 3, list all phonetic symbols it corresponds to and their mapping probabilities
   a. For each phonetic symbol, add the mapping probabilities of symbols in the similar-sound list
5. Check which symbols from step 4 are valid children of the current node and their probabilities as a child of the current node
   a. For each phonetic symbol, add the child probabilities of symbols in the similar-sound list
6. If the letter prefix is in the graphemic substitution list, replace by the substitution symbol and repeat steps 4 and 5
7. If there are vowel-sound symbols as unvisited children of the current node, assume they are abbreviated sounds [3]
   a. For each phonetic symbol, get the child probability, and set a dampening constant for the mapping probability.
8. Generate new nodes:
   a. Remove letter prefix from word
   b. Add phonetic symbol to phonetic string
   c. Multiply node probability by mapping probability & child probability
   d. Update current node to the child node with the correct phonetic symbol

**TABLE I. CORPUS COMPOSITION AND TEST RESULTS BY TYPE.**

<table>
<thead>
<tr>
<th>Normalization type</th>
<th>Word count</th>
<th>Top-1 accuracy</th>
<th>Top-5 accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall Average</td>
<td>2608</td>
<td>30.2%</td>
<td>59.7%</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>806</td>
<td>29.0%</td>
<td>52.6%</td>
</tr>
<tr>
<td>Phonetic substitution</td>
<td>130</td>
<td>53.8%</td>
<td>78.5%</td>
</tr>
<tr>
<td>Graphemic substitution</td>
<td>58</td>
<td>29.3%</td>
<td>58.6%</td>
</tr>
<tr>
<td>Stylistic variation</td>
<td>820</td>
<td>31.6%</td>
<td>65.7%</td>
</tr>
<tr>
<td>Letter repetition</td>
<td>641</td>
<td>28.1%</td>
<td>62.2%</td>
</tr>
<tr>
<td>Multiple types</td>
<td>153</td>
<td>17.9%</td>
<td>38.4%</td>
</tr>
</tbody>
</table>

**IV. CONCLUSION**

In this paper, we presented a novel normalization algorithm for the OOV words commonly found in microtext messages based on the idea that these words will phonetically similar to their IV word counterparts. Our prototype system computes the probable phonetic reading of words based on their spelling using training examples, and then determines the likely IV English equivalents. Our results show that the system performs adequately, although there is clear room for improvement.

**REFERENCES**


