Word Normalization Using Phonetic Signatures

Vincent Jahjah[1], Richard Khoury[2(✉)], and Luc Lamontagne[1]

[1] Department of Computer Science and Software Engineering,
Laval University, Quebec City, QC, Canada
vincent.jahjah.1@ulaval.ca, luc.lamontagne@ift.ulaval.ca
[2] Department of Software Engineering, Lakehead University, Thunder Bay, ON, Canada
richard.khoury@lakeheadu.ca

Abstract. Text 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 detail a new word-searching strategy based on the idea of sounding out the consonants of the word. We describe our algorithm to extract the base consonant information from both miswritten and real words using a spelling and a phonetic approach. We then explain how this information is used to match similar words together. This strategy is shown to be time efficient as well as capable of correctly handling many types of normalization problems.

Keywords: Social media · Normalization · Wiktionary · TheFreeDictionary

1 Introduction

Social network messages have become omnipresent in today's world, and their highly relaxed spelling rules and tolerance to extreme irregularities in spelling poses problem when trying to apply traditional Natural Language Processing (NLP) tools and techniques, which were developed for properly-written English text. A sampling of Twitter messages studied in [1, 2] found over 4 million out-of-vocabulary (OOV) words, and, new spelling variations are created constantly, both voluntarily and accidentally.

The challenge of developing algorithms to automatically correct the OOV words found on social media and replace them with the correct in-vocabulary (IV) words is known as normalization. In [3], it was suggested that OOV words are often pronounced similarly to the IV words they represent; for example, "2niite" sounds almost exactly like "tonight". In this paper, we further add the intuition that this similarity is due to the consonants used, rather than the vowels. We propose to make an algorithm that matches OOV words to IV words using the pronunciation of their consonants.

2 Background

Today, billions of written messages are sent online. A significant proportion of them are written in an informal manner with little to no concern for proper spelling. This has given rise to the challenge of normalization, or of developing algorithms to determine which real IV words are intended by the miswritten OOV words found on social

R. Khoury and C. Drummond (Eds.): Canadian AI 2016, LNAI 9673, pp. 180–185, 2016.
DOI: 10.1007/978-3-319-34111-8_23

networks. While the variety of spellings of OOV words is seemingly endless, it was noted in [2] that their creation follows a small set of simple rules. The rules proposed in [2] are *abbreviation* (deleting letters from the word, e.g. "together" → "tgthr"), *phonetic substitution* (substituting letters for other symbols that sound the same, e.g. "2gether"), *graphemic substitution* (substituting a letter for a symbol that looks the same, e.g. "t0gether"), *stylistic variation* (misspelling the word to make it look like one's personal pronunciation, e.g. "togeda", "togethor"), and *letter repetition* (e.g. "togetherrr"). Approaches to normalize the OOV words are varied, and can include various degrees of automatic data gathering [2–4], OOV pre-processing [5], hand crafted data [6, 7], and considerations for the OOV word's context [4, 7]. Some works focus on specific types of normalizations ([8] for letter repetitions, [9–11] for letters deletion).

The system proposed in [7] used algorithms to generate a list of 660,000 OOV versions of 47,000 words, and performed a lexical and syntactic analysis of the messages in order to recognize observed words. In [2], the authors developed an algorithm to discover the most common letter substitutions between pairs of IV and OOV words and compute their probabilities. They then use their probabilistic model to generate the most probable IV words of new OOV words. In [9], the authors used a rule-based algorithm to map between IV and OOV words by removing double letters, unnecessary vowels, and prefixes and suffixes. In [10], they improved on their original idea by training a probabilistic machine translation model instead of using rules.

3 Methodology

The fundamental assumption of our work is that consonants contain more of the phonetic information of the words than vowels do. As a result, we believe OOV words can be more easily recognized if one ignores vowel differences but gets the consonants right. Compare for example the ease of recognizing "byutfl" or "btfl" as variants of "beautiful", as opposed to "eauiu". The core aspect of the methodology we propose is the extraction of the phonetic information of each word, which we call the signature of the word. We further distinguish between two types of signatures, namely the visual signature (VS) which is what the sequence of consonants should look like when written down, and the phonetic signature (PS), which is what the sequence of consonants should sound like when read out loud. For example, the VS of the word "accent" would be "cnt" and its PS would be "Xnt". These signatures are the information that our methodology will then use to match OOV words to IV English words. For example, the OOV words "accnt" and "aksnt" will both have the same signatures as 'accent', and can thus be matched to that IV word.

3.1 Phonetic Data and Signature Extraction

The first stage of our methodology consists in extracting the VS and PS of real English words, to serve as a reference. The algorithm also extracts simplified reconstructions of the word with vowels reinserted at the correct places. For instance, the word "accent"

has a visual reconstruction (VR) of "acent" and the phonetic reconstruction (PR) "aXent". Reconstructions will be useful in the word matching stage.

Our algorithm starts with a training data set of English words paired with their IPA spellings, which represent their pronunciations. These pairs can be obtained from online dictionaries, such as Wiktionary and TheFreeDictionary, from which we obtained 78,327 pronunciations for 48,864 words. Steps 1 and 2 of the algorithm extract the VS and VR of each word. The VS is the sequence of consonants in the word as it is spelled, with all vowels and duplicates removed ("accent" → "cnt"). The VR is the VS with vowels included ("accent" → "acent"). They can thus be created together. Step 3 simplifies the IPA version of the word by merging common sets of single sounds into a single symbol. This is done by finding and replacing the following six pairs of IPA symbols: tʃ → C, ks → X, kw → Q, hw → w, tz → Z, and gz → G. In step 4, this simplified IPA is used to create the PS in the same process as in steps 1 and 2, namely by removing duplicate IPA symbols and vowel IPA symbols in our data set (the ambiguous IPA symbols "j" and "w" are treated as consonant sounds for now). Step 5 and 6 create the temporary VS and PS by replacing vowels and vowel IPAs with * in the VR and the simplified IPA respectively (e.g. "one" → "*n*").

Step 7 requires matching consonant letters in the word to sounds in the IPA string. One consonant can be matched to an identical IPA symbol (e.g. "too" → "tu") or to a different IPA symbol ("act" → "akt"). But an IPA symbol might not have a matching consonant (the "j" symbol in "dupe" → "djupe"), or a consonant in the word might not be matched to an IPA symbol (the letter "b" in "subtle" → "sutl"). We use the temporary signatures from steps 5 and 6 to clarify matters (for example "one" → "won" has the temporary signatures "*n*" → "w*n", indicating that a consonant sound occurs before the first vowel without an associated letter). Our algorithm uses a set of about 100 rules and 200 exceptions to handle our 78,000 word-IPA pairs.

3.2 Signature Generation of OOV Words

The second major stage of our methodology consists in generating the signatures of a new OOV word, for which IPA information is unknown. We first account for three of the five main types of word transformations. For the repetition type, we reduce all repeated letters to doubles and singles ("suuuuppppp" → "suupp", "sup"). While the IV signatures actually discard duplicates, the matching algorithm in the next stage will prioritize exact matches, so it is important to generate both. For graphemic and phonetic substitutions, we substitute the numbers for letters. When multiple substitutions are possible (1 could stand for L, I, or "one"), the algorithm lists all possible cases. The output will be a list of variations of the OOV word without digits and with limited repetitions ("1337" → "oneet", "leet", "ieet", "onet", "let", "iet").

Next, we consider four ways of generating the signatures of a word. One of these will result in the correct signature for the IV word. First, the VS and VR of each OOV word in the list of variations is generated using steps 1 and 2 of the extraction algorithm. Next, the PS and PR are generated, using an approximate IPA pronunciation generated from the set of rules from step 7 of the extraction algorithm. Third, to deal with the possibility that there are permutations of consonants in the word ("answer" → "asnwer"),

our algorithm generates VS and VR pairs of permutations of adjacent consonants ("asnwers" → "answers", "asnwesr"). Finally, we deal with potential suffixes in the OOV word. These suffixes are often altered in standard ways, which means most of them can be easily corrected using nine simple rules: in → ing, a → er, a → or, z → s, n → ing, as → ers, as → ors, d → ed, and ah → er. We remove the suffix ("dansa" → "dans") and make a VR and VS for the root.

3.3 Similar Word Matching

The third and final stage of our methodology consists in determining which IV English word the OOV word sounds like. First we find a set of IV words with identical signatures, along with their occurrence probability (taken from standard word frequency tables such as ProjectGutenberg). Then our algorithm applies a set of heuristics to score the IV as a match to the OOV word. The first heuristic considers whether the signature is visual, phonetic, permuted, or suffixed, corresponding to the four ways of generating the signature in part 3.2. The matches are prioritized in that order. The second heuristic considers on how the first letter of the matching word compares to that of the OOV word. The highest value is given if the first letter of both words is the same, then a lesser value is given, in order: if the IV word begins by a silent vowel truncated in the OOV word ("nough" → "enough"), if an initial letter H was deleted, if the initial vowel was phonetically transformed ("ate" → "eight"), and finally if a non-silent initial vowel was deleted. In any other case, the word is eliminated from consideration. The final heuristic scores the IV word compared to the OOV word's reconstruction. Maximum score is given if the IV word is identical to the reconstruction, then if there is only a suffix difference. Failing that, the algorithm computes the Levenshtein distance between the IV word and reconstruction, using an implementation of the distance that ignores duplicate letters. Two lesser scores are given if the distance is 0 or 1, respectively. If the distance is greater than 1, then no score is given.

4 Experimental Results

In order to test our normalization methodology, we used the test set from [2, 4]. This corpus lists 3525 OOV words that were observed in real tweets, along with their corresponding IV English words, sorted into seven basic types [2], namely abbreviation, phonetic substitution, graphemic substitution, stylistic variation, letter repetition, typographical errors (a default catch-all category), and multiple types at once.

Table 1 gives the accuracy of our algorithm, both overall and per type of normalization problem. A top-n match is defined as the presence of the correct IV word in the n most probable IV words found by our algorithm. The results include only the 3525 OOV words mapping to known IV words in our system.

One thing to note is that the overall top-2 accuracy increases by over 10 % compared to the top-1. This may be due to the ambiguity inherent to OOV spellings. As mentioned earlier, the OOV word "accnt" could map to either "accent" or "account" (incidentally

the top two IV words returned for that word), and it is impossible to tell which one the user intended without the sentence context, which is not taken into account in our work.

Table 1. Corpus composition and test results by type.

Type	Words	Top-1	Top-2	Top-5	Top-10
Overall Average	3525	56.6 %	66.1 %	73.6 %	77.2 %
Abbreviation	658	30.8 %	41.4 %	53.1 %	57.2 %
Phonetic substitution	60	66.6 %	76.6 %	86.3 %	93.3 %
Graphemic substitution	63	80.9 %	80.9 %	80.9 %	80.9 %
Stylistic variation	1646	55.7 %	67.4 %	76.7 %	81.3 %
Letter repetition	756	92.1 %	96.8 %	98.5 %	99.2 %
Typo	141	17.0 %	21.9 %	23.4 %	25.5 %
Multiple types	201	32.8 %	43.7 %	51.2 %	56.7 %

Repetition is the type of normalization problem that our system handles best, reaching a top-1 accuracy of 92.1 %. Since our system is designed to sound out consonants, we likewise find that abbreviation-type OOV words that crop vowels, silent letters, and consonants in multi-letter sounds cause no problems. Nearly all of the abbreviation-type OOV words that failed to be recognized were missing voiced consonants. Phonetic substitutions are also handled very well by our system. The main failure case occurs when the substitution affects the first letter of the word. These cases are blocked by the second heuristic of our matching algorithm.

Graphemic substitutions have the unusual behaviour of being only matched in top-1 or not at all. This can be explained by the fact that all substitution cases we accounted for will result in exact signatures and perfect IV word matches, while most substitutions that were not included in our system (such as the letter Q replacing a G) will lead to wrong OOV word signatures that cannot be recognized.

The final normalization type is typos. These include problems such as letters permutations or accidentally repeated syllables (“novemember” → “November”), which our system can handle relatively well, or deliberately using foreign words or slang (“oogie” → “disgusting”), which our system fails completely on. This type of normalization problem violates our fundamental assumption, that OOV words are phonetically-similar to the intended IV ones.

5 Conclusion

In this paper, we presented a novel normalization algorithm for the OOV words commonly found in social media messages. The two fundamental intuitions behind our algorithm are that these OOV words are phonetic approximations of the IV words they represent, and that consonants are more phonetically important than are vowels. Consequently, our algorithm is designed to extract the consonant signature of the OOV word, and to match it with IV words that have identical signatures, ranking the matches using a set of linguistic transformation heuristics. Our results show that this method can recognize 80.0 % of OOV words that appeared in a testing corpus of Twitter messages,

with 56.6 % of them ranking the correct IV word in first place. Future work may look at handling these special cases, for example by integrating spell-checking routines to deal with common typos or lists of common abbreviations such as in [7].

References

1. Petrovic, S., Osborne, M., Lavrenko, V.: The Edinburgh Twitter corpus. In: Proceedings of the Naacl Workshop on Computational Linguistics in a World of Social Media, Los Angeles, USA, pp. 25–26 (2010)
2. Liu, F., Weng, F., Wang, B., Liu, Y.: Insertion, deletion, or substitution?: Normalizing text messages without pre-categorization nor supervision. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, vol. 2, Stroudsburg, USA, pp. 71–76 (2011)
3. Khoury, R.: Phonetic normalization of microtext. In: Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, 25–28 August 2015, Paris, France, pp. 1600–1601
4. Liu, F., Weng, F., Jiang, X.: A broad-coverage normalization system for social media language. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, Jeju, Korea, pp. 1035–1044 (2012)
5. Clark, E., Araki, K.: Text normalization in social media: progress, problems and applications for a pre-processing system of casual english. In: PACLING 2011. Procedia - Social and Behavioral Sciences, vol. 27, pp. 2–11 (2011)
6. Han, B., Cook, P., Baldwin, T.: Lexical normalization for social media text. ACM Trans. Intell. Syst. Technol. (TIST) **4**(1), article no. 5. Digital Publication (2013). http://dl.acm.org/citation.cfm?id=2414425&picked=prox&CFID=768981160&CFTOKEN=83762437
7. Jose, G., Raj, N.S.: Lexico-Syntactic Normalization Model for noisy SMS Text. Dept. of Comput Schi., SCMS Sch. of Eng. & Technol., Ernakulam, India, November 2014
8. Hirankan, P., Suchato, A., Punyabukkana, P.: Detection of wordplay generated by reproduction of letters in social media text. In: 10th International Joint Conference of JCSSE, pp. 6–10, May 2013
9. Pennell, D.L., Liu, Y.: Normalization of text messages for text-to-speech. In: Proceedings of the 35th International Conference on Acoustics, Speech and Signal Processing, Dallas, USA, pp. 4842–4845 (2010)
10. Pennell, D.L., Liu, Y.: Normalization of informal text. Comput. Speech Lang. **28**(1), 256–277 (2014)
11. Maitama, J.Z., et al.: Text normalization algorithm for facebook chats in hausa language. In: 5th International Conference of ICT4M, pp. 1–4, November 2014