Learning the costs for a string edit distance-based similarity measure for abbreviated language

Laura Alonso i Alemany

NLP group
FaMAF-UNC
Córdoba, Argentina
alemany@famaf.unc.edu.ar

Abstract. We present work in progress on word normalization for user-generated content. The approach is simple and helps in reducing the amount of manual annotation characteristic of more classical approaches. First, ortographic variants of a word, mostly abbreviations, are grouped together. From these manually grouped examples, we learn an automated classifier that, given a previously unseen word, determines whether it is an ortographic variant of a known word or an entirely new word. To do that, we calculate the similarity between the unseen word and all known words, and classify the new word as an ortographic variant of its most similar word. The classifier applies a string similarity measure based on the Levenshtein edit distance. To improve the accuracy of this measure, we assign edit operations an error-based cost. This scheme of cost assigning aims to maximize the distance between similar strings that are variants of different words. This custom similarity measure achieves an accuracy of .68, an important improvement if we compare it with the .54 obtained by the Levenshtein distance.

Keywords: Natural Language Processing, String Edit Distances

1 Introduction and Motivation

In recent times there has been a significant increase in informal user-generated text where users make heavy use of personalized words: abbreviations, acronyms, skipping vowels, using numbers instead of letters, etc. Good examples of that are community-centered blogs, short text messages by mobile phones, user states in social networks, advertisements or auction listings.

This user-generated content provides a growing amount of privileged information for a variety of goals, ranging from product or service reviews to epidemic surveillance. Therefore, being able to automatically process this kind of text is both promising and necessary. Promising because of the rich information it carries, very often, information that cannot be found elsewhere. Necessary because the amount of user-generated text is big and growing and also different from

standard language and rapidly changing, thus making it unfeasible to treat it manually.

Big amounts of texts are usually pre-processed with standard Natural Language Processing (NLP) tools to obtain some linguistic abstraction on the raw text. However, standard NLP tools cannot be applied directly to user-generated text because it presents many differences with respect to the standard dialect of languages, at lexic, syntactic and even semantic levels. This makes it very difficult to automatically extract information from them or apply any other kind of more complex automatic treatment, like machine translation [2] or to feed a database [10].

A usual strategy for the automatic treatment of these messages is to turn them (or "translate" them) into their equivalents in the standard dialect. Those translated versions can be successfully treated by standard tools for Natural Language Processing. An added value for this strategy is that it provides information about the relationship between the standard dialect and upcoming variants. On top of this, if we use unsupervised machine learning methods, we can automatically incorporate new messages in the strategy, and thus keep pace with the rapid evolution of new variants.

The first necessary task to apply NLP tools is to normalize the vocabulary in these applications. Very often, vocabulary normalization for this kind of text is done by hand-made dictionaries of abbreviations and variants[1]. However, the variation in this kind of text is very high, and new variants of words and entirely new ways of expression are produced rapidly. In this context, a more automated approach becomes necessary to keep pace with the evolution of language.

In this paper we apply an edit-distance based method to normalize non-standard words. Our edit distance must be able to differentiate between discriminating and non-discriminating edit operations. Not all edit operations are equal and not all contexts are equal for an edit operation to take place. For example, common vowels can be deleted with less cost, or deletions of consonants are less costly if they are within a group of consonants than surrounded by vowels, with a less prominent role in shaping the syllable. Although linguistic insights can be useful to assign weights for this kind of phenomena, their number, variety and the rapid evolution of word forms makes a machine learning approach more adequate.

We develop an approach to alleviate human intervention in the process of normalization of user-generated text. We substitute dictionary-based normalization by clusters of words with the same canonical form. Then, given a new word, an automatic classifier finds the most similar word in the manually created clusters. If similarity between these two strings is over a given threshold, the new word is clustered together with the most similar; if not, it constitutes a new cluster. In this setting, the task of the human annotator is reduced to creating an initial set of clusters and validating the associations created by the classifier.

[1] Many on-line services for normalization of texting language resort to dictionary-based strategies, like `http://transl8it.com/`, `http://www.lingo2word.com/translate.php`, `http://www.dtxtrapp.com/`.

Grouping together variants of the same word seems like a good approach to normalization, in contrast with assigning a canonical form to all variants of a word. Then, the task of normalizing changes from finding similarities between an unknown word and canonical forms to finding similarities between an unknown word and all known variants of a word. This allows to capture variants of variants, which occur often in the dynamic context of user-generated content.

In this paper we will focus in newspaper advertisements but we expect that our method can be easily extended to other domains with similar phenomena. Advertisements present the advantage of providing plentiful, time-stamped corpora, as opposed to other genres like short text messages.

The rest of the paper is organized as follows. First, we review some of the previous work on normalization of highly abbreviated text and on learning edit distances. Then, we describe the data and method that we use to normalize advertisement text. In Section 4 we go through the different approaches to modify the costs of edit operations in a string edit distance to improve the accuracy of the similarity measure between words. Experiments and results are presented in Section 5, followed by an analysis of the linguistic implications of the costs learned for the edit distance. We conclude outlining some extensions of this work.

2 Previous Work

Normalization of text is a crucial task for many NLP applications, specially in genres where new variants of words are produced rapidly, so that a lexicon becomes too static, failing to cover bigger proportions of text. That is the case of sms and their abbreviations [2, 4, 1, 7, 5] or medical text, with a heavy use of acronyms [13, 17, 11, 16]. Normalization and edit-distance based approaches to normalization can also be found in application areas like spelling correction or speech recognition.

Sproat *et al.* [15] present an extensive work on normalization of non-standard words in different genres and applying both supervised and unsupervised techniques. The approaches they present rely heavily on heuristic approaches to the most systematic forms of variation in words, leaving learning for the more free parts. This approach proves successful, but relies on a signifcant amount of handmade rules that have to be updated regularly if they are to keep pace with the current pace of change in user-generated content. We aim to develop a method that reduces the amount of human intervention required.

Our hypothesis is that a convenient measure of similarity between words should be able to group words together automatically. String edit distance seems specially adequate for this purpose because we are trying to assess whether a given string is a different form of writing another string. However, general purpose distances lose accuracy in very short strings, as is the case in abbreviations. In this context, a finer-grained edit distance is necessary, a distance where the discriminating power of different edit operations is not homogeneous. Thus, the problem of finding a finer-grained edit distance becomes the problem of finding the costs of edit operations that discriminate best.

Several approaches have been proposed to learn the costs of edit operations for string edit distances, ranging from stochastic transducers [14, 3, 12] to conditional random fields [9], maximum entropy approaches [13], noisy channel models [2, 5] or pair-Hidden Markov Models [6].

We propose a simple model that does not have the expressive power (and the correspondingly huge search space) of more complex models like [9], but achieves a good performance in the task at hand, comparable to other approaches. For example, [4] use a Hidden Markov Model trained on a corpus of SMS manually aligned with their standard English transcriptions, and achieve an accuracy of 57.7%. Using an unsupervised approach in a comparable task, [5] obtain an accuracy of 59%.

3 Data and Method

Our approach to normalization of advertisement text is as follows. First, we manually group together ortographic variants of the same word in a sample of the corpus. Then, for a new word w, we determine whether it is an ortographic variant of a known word or a new word.

We detail both these steps in what follows.

3.1 Manually classified examples

We are working with a corpus of 1 million words of Spanish classified ads in the real estate category, from the Argentinian local newspaper *La Voz del Interior*[2].

We have manually grouped together the ortographic variants of words found in a single day of advertisements, totalling 3359 ads. Out of 55946 token words (blank-separated strings), 8727 unique words were found, leaving numbers out. We manually clustered words in 2824 groups, of which 113 had 10 or more words and 1700 were singletons. We also created a smaller corpus to carry out smaller experiments, with words occurring ten times or more, totalling 927 words, with 493 classes, of which 333 were singleton classes and 5 had 10 or more elements.

Flexive variants of the same root are considered in the same class, because they tend to abbreviate to the same abbreviation (e.g.: "*Ot. ot. ot ots Ot otr otra otros.*", with different flexive forms of "(an)other" and abbreviations that are valid for all of them).

Words were not separated from punctuation, because in this context punctuation is ambiguous, it can signal abbreviation but also relationship between two parts of a single word. We did not separate words in multiword expressions, many of which are written together (e.g.: "*3d*" for "*3 dormitorios*", "three bedrooms").

[2] `http://www.clasificadoslavoz.com.ar/`

3.2 Classification of unseen words

For a previously unseen word w, we find the word c in the manually grouped sample that is most similar to w. If the similarity between w and c is over a given threshold, then w is assigned to the same group as c, as an ortographic variant of the same word. On the contrary, if w and c are too dissimilar, a new group is created with the new word w. We measure similarity between words using the inverse of a string edit distance: the smaller the distance, the bigger the similarity.

The basic edit distance used to calculate the similarity between words is Levenshtein distance [8]. This distance counts the number of changes that are necessary to transform one string into another. Each change, or edit operation, has a uniform cost of 1, while leaving the same character in both string has a cost of 0. Edit operations are applied from left to right in the string.

We have introduced some modifications in the basic Levenshtein distance, trying to improve its accuracy in finding ortographic variants.

First, we have weighted the costs of edit operations, which are uniform in Levenshtein distance. As developed in Section 4, we have explored two strategies to determine the cost of edit operations that reflect their impact to identify ortographic variants of a word, as observed in manually classified examples.

Second, edit operations have been enriched with context, taking into account the character before and after the character where the operation takes place. So, different weights were assigned to the same edit operation if it occurred with different characters before or after it.

In order to avoid the data sparseness that comes together with a higher granularity in edit operations, we assigned weights to edit operations with full context (with one character before and one character after it), with partial context (with the caracter before and the character after it as separated instances) and without context (without any characters surrounding it).

Then, when finding the edit distance for a pair of strings, edit operations were applied using a back-off strategy, in a more- to less-specific order: if we had learned a weight for the operation with the left and right context, we applied that, if not, we applied the weight with partial context to the left or to the right, if we didn't have evidence for neither of those contexts, we applied the weight of the operation without any context. In case the edit operation had not been seen in the training corpus, it would have no weight. In that case, the Levenshtein distance is applied, with cost 0 in case of match between characters and cost 1 for mismatch, insertion or deletion.

4 Learning string similarity measures based on edit distance

In the context of newspapers advertisements, as is the case in other contexts with many abbreviations, like sms or scientific papers, we find many very short strings, with possibly very high variations between them, for example *"l/c liv/com.*

liv/com liv-com, liv/com, l/c. lc. lc lc, l/com. l/com L/c l/c," for "living-dining room". With uniform edit costs, most of the shortest strings will be equally distant to many different words. We need our edit distance to capture degrees in the importance of edit operations, assess which make a meaningful difference in strings and which don't.

Taking Levenshtein distance as a starting point, we tried to find a similarity measure that improved its performance by modifying the cost associated to the different edit operations. Costs were modified by exploiting evidence from our manually annotated corpus, in two different ways: by a combination of random and best-first search in the space of costs associated to each edit options, and by associating error-driven costs to edit operations. We develop these approaches in what follows.

4.1 Random search of the space of costs

As a first approach, we did a random search to find a configuration of costs that would improve the baseline edit distance. Since the search space is very big, we explored changing the values only for those operations that actually occurred when the words in the corpus were aligned with each other. We used a combination of random and best-first search in the space of costs of edit operations.

The procedure worked as follows. First, we took 50 edit operations at random, and modified their initial cost by first adding 1 and then substracting 1 to it. For each modification in each operation, we evaluated the impact on accuracy in a small random sample of the corpus, and recorded the obtained accuracy. Then, we ordered operations in decreasing order of accuracy, and evaluated the impact of the modification in a bigger sample of the corpus.

If accuracy in this bigger sample was bigger than with the Levenshtein distance, we introduced the modification in the running set of costs for edit operations, and proceeded to evaluate the following modification. We stopped evaluating modifications with bigger samples of the corpus when the accuracy obtained in the smaller sample was smaller than the best accuracy obtained so far. Then, we started the search again by exploring another 50 random operations, until a significant improvement was achieved.

4.2 Error-driven search

In order to improve the accuracy of the Levenshtein distance more efficiently than with random search, we applied an error-driven schema to modify edit operation costs.

We assigned edit operations a cost obtained from the number of times that the edit operation was seen in the alignment of a word with the one found most similar to it, the one that would be identified as an ortographic variant of the same word. Alignments were obtained by applying the Levenshtein distance to the pairs of words. Then, we counted the times that the edit operation was seen in an alignment of a pair of words that were actually variants of the same word (match), and different words (mismatch).

Then, the value assigned to each edit operation was the proportion of times it occurred in a mismatch minus the proportion of times it occurred in a match, so that operations occurring more often in matches received lower, even negative costs. This minimizes the cost of aligning words with edit operations that occurred mostly in matches, and maximizes the cost of edit operations occurring mostly in mismatches.

$$cost_{eo} = \frac{mismatch_{eo}}{N_{eo}} - \frac{match_{eo}}{N_{eo}}$$

In order to avoid inaccurate estimates from events occurring too few times, we only took into consideration those operations occurring more than 10 times in the manually annotated examples. Moreover, we took 4 samples of the corpus and found the cost assigned to the operation in each of the 4 samples, and we only incorporated costs whose standard deviation in the four samples was smaller than three times the mean of the cost. In those cases, the cost of the edit operation was the average cost in the four samples.

Again, when seeing an edit operation for which we had no cost recorded, we backed off to the Levenshtein distance.

5 Experiments and Results

We evaluated the above mentioned modifications using the manually annotated corpus mentioned in Section 3.1. We evaluated accuracy as the proportion of words in the evaluation corpus that were correctly matched with an ortographic variant of the same word or singled out as the only way found in the corpus to write a given word. The threshold to discriminate different words was set to 3, that is, a new group was created for those words whose closest candidate was at an edit distance bigger than 3.

The Levenshtein distance obtained an accuracy of .54 in the 900-word corpus.

The random search on the space of edit operation costs did not yield significant improvements with respect to the Levenshtein distance. Indeed, for most of the samples, the accuracy using Levenshtein distance was the same as using the modified costs.

The error-driven cost modification was more successful than the random search. When trained on the 900-word corpus, the accuracy raised to .58. However, few (376) edit operations were modified because most of them did not meet the conditions to be taken into account: either too few occurrences or variability too high across samples. When trained on the 8000-word corpus, many more (6324) edit operations met the conditions, which yielded an accuracy of .68 on the 900-word corpus.

6 Qualitative Analysis of Results

We manually inspected the weights learned for the parameters of edit distances. This allowed to find regularities to support linguistic hypotheses about how

users introduce variations in words, while keeping them understandable for fellow readers.

First of all, we found that the whole approach was working as expected because identical characters were indeed assigned lower weights than different characters, and that lower-case and upper-case variants of the same character were also assigned lower weights. With respect to single characters, parentheses were the ones with lower alignment costs, together with the + sign.

At the opposite side, the single characters that were assigned highest costs were numbers (specially 2 and 3), and letter 's', specially in the end-of-word context. These three characters have the common property that they convey cardinality, which is a very important piece of meaning in the context of real estate context, and users do not want to risk being misunderstood.

Weights were much higher for mismatches with unfrequent consonants, for example, between *z* and other consonants, or between letters sounding very differently, like between *t* and other letters. Insertions and deletions obtained lower weights than mismatches. The most penalized insertions were those that would re-create a syllabic structure, that is, insertion of a vowel between consonants or insertion of consonants at the end of a word.

7 Conclusions and Future Work

We have presented work in progress aiming at the normalization of words in classified advertisements. Our approach requires less human intervention than dictionary-based approaches. Ortographic variants of the same word are grouped together, and the longest form is taken as the canonical. Then, texts are normalized by substituting every form by its canonical form.

An automatic classifier assigns each new word to a pre-existing group of ortographic variants or establishes a new group for the new word. Similarity between words is calculated by an adaptation of the Levenshtein distance, where costs of edit operations are weighed by their occurrences in errors or matches of the classifier. This tailored edit distance, with costs of edit operations learned from manually classified examples, improves on the Levenshtein distance increasing accuracy from .54 to .68.

As future work we plan to compare the performance of our learned distance with the Jaro-Winkler distance [18], which is specially adequate to deal with abbreviations by its special treatment of preffixes.

We will also increase the size of context to take into account for edit distance to apply, especially trying to capture the proximity to the beginning and end of the word.

We are planning to carry out a large-scale evaluation of the temporal dimension in the evolution of abbreviations. We will compare the performance of the classifier if new words are provided as they appear in the newspaper, incorporating words in a daily basis and then using all the incorporated words to calculate distances with the words in the following day, or if we try to classify all words in the corpus at the same time.

We also want to evaluate the performance of the costs of edit operations learned in the domain of advertisements to detect ortographic variants of words in other domains. More concretely, a similar approach for short text messages is underway, including the construction of a corpus of messages for Spanish, covering the Argentinian and Uruguayan dialect.

Acknowledgements

This research has been partially funded by project *Representation of Semantic Knowledge* TIN2009-14715-C04-03 of the Spanish Ministry of Education and Culture, and by project PAE-PICT-2007-02290, funded by the National Agency for the Promotion of Science and Technology in Argentina.

References

1. Acharyya, S., Negi, S., Subramaniam, L.V., Roy, S.: Unsupervised learning of multilingual short message service (sms) dialect from noisy examples. In: AND '08: Proceedings of the second workshop on Analytics for noisy unstructured text data. pp. 67–74. ACM, New York, NY, USA (2008)
2. AiTi, A., Min, Z., PohKhim, Y., ZhenZhen, F., Jian, S.: Input normalization for an english-to-chinese sms translation system. In: The Tenth Machine Translation Summit (2005)
3. Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: Proceedings of the ninth ACM SIGKDD (2003)
4. Choudhury, M., Saraf, R., Jain, V., Mukherjee, A., Sarkar, S., Basu, A.: Investigation and modeling of the structure of texting language. Int. J. Doc. Anal. Recognit. 10(3), 157–174 (2007)
5. Cook, P., Stevenson, S.: An unsupervised model for text message normalization. In: Workshop on Computational Approaches to Linguistic Creativity. NAACL HLT 2009 (2009)
6. Durbin, R., Eddy, S., Drogh, A., Mitchison, G.: Biological sequence analysis: Probabilistic models of proteins and nucleic acids. Cambridge University Press (1998)
7. Kobus, C., Yvon, F., Damnati, G.: Normalizing sms: are two metaphors better than one? In: COLING '08: Proceedings of the 22nd International Conference on Computational Linguistics. pp. 441–448. Association for Computational Linguistics, Morristown, NJ, USA (2008)
8. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Cybernetics and Control Theory 10(8), 707–710 (1966), original in *Doklady Akademii Nauk SSSR* 163(4): 845–848 (1965).
9. McCallum, A., Bellare, K., Pereira, F.: A conditional random field for discriminatively-trained finite-state string edit distance. In: Proceedings of the Proceedings of the Twenty-First Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-05). pp. 388–395. AUAI Press, Arlington, Virginia (2005)
10. Michelson, M., Knoblock, C.A.: Phoebus: a system for extracting and integrating data from unstructured and ungrammatical sources. In: AAAI'06: proceedings of the 21st national conference on Artificial intelligence. pp. 1947–1948. AAAI Press (2006)

11. Okazaki, N., Ananiadou, S., Tsujii, J.: A discriminative alignment model for abbreviation recognition. In: COLING '08: Proceedings of the 22nd International Conference on Computational Linguistics. pp. 657–664. Association for Computational Linguistics, Morristown, NJ, USA (2008)
12. Oncina, J., Sebban, M.: Learning stochastic edit distance: Application in handwritten character recognition. Pattern Recognition 39(9), 1575–1587 (2006)
13. Pakhomov, S.: Semi-supervised maximum entropy based approach to acronym and abbreviation normalization in medical texts. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL). pp. 160–167 (2002)
14. Ristad, E.S., Yanilos, P.N.: Learning string edit distance. IEEE Transactions on Pattern Analysis and Machine Intelligence 20, 522–532 (1998)
15. Sproat, R., Black, A., Chen, S., Kumar, S., Ostendorf, M., Richards, C.: Normalization of non-standard words. Computer Speech and Language 15(3), 287–333 (2001)
16. Stevenson, M., Guo, Y., Al Amri, A., Gaizauskas, R.: Disambiguation of biomedical abbreviations. In: BioNLP '09: Proceedings of the Workshop on BioNLP. pp. 71–79. Association for Computational Linguistics, Morristown, NJ, USA (2009)
17. Torii, M., Liu, H., Hu, Z., Wu, C.: A comparison study of biomedical short form definition detection algorithms. In: TMBIO '06: Proceedings of the 1st international workshop on Text mining in bioinformatics. pp. 52–59. ACM, New York, NY, USA (2006)
18. Winkler, W.E.: The state of record linkage and current research problems. Tech. Rep. Internal Revenue Service Publication R99/04, Statistics of Income Division (1999)