Machine Learning approach for Domain Specific Spell Correction

Suraj S Nair, Sreeram, Prashant Gupta, Muthu Arvind, Aditya Bhandare, Kumar Vaibhav, Sai Sathya Bapuji

***Abstract*- The presented work illustrates the use of machine learning techniques, specifically Regression and Neural Networks, combined with novel use of string comparison algorithms for isolated word correction as applied to the medical domain, or any other domain for that matter. We have used edit distance, dice coefficient, regular expression, 'headers' technique and lig3 to train the model. We also present the evaluation of these techniques on medical domain as a proof of concept. The controlled experimental results combined with our unique use of features showed that this new method with neural networks is significantly better than linear regression and is also a proof that domain specific word correction task should be dealt with differently.**

Index Terms – Machine Learning, Training set, Dictionary, Measures of similarity, Edit distance, Linear regression, error/cost function, Neural Networks.

I. INTRODUCTION

It is a well-known fact that medical terms are often hard to spell. In such a situation, a spell corrector specifically for the medical domain proves to be a very handy tool for anyone dealing with such text – be it for analytical usage in the area of text mining or for end users.

The implication of errors arising as a result of misspelling a medical word is immense. Reports show that there is a significant error rate observed in patient records. An intelligent system could provide the users with some suggestions in order to assist in using correct words when dealing with medical information. In the medical domain, spelling correction as a problem area is not new. Spelling correction has been applied to chief complaint data to expand acronyms, abbreviations and truncations and to correct spelling errors. The rapid advances in computer hardware and software also mainstreamed the use of spell checkers in software development as well.

In the present work, we focus on isolated term correction rather than contextual term correction.

Spell correction, in general, is not a new problem and many approaches have been taken to solve it. Typically, these approaches use measures of similarity to identify the closeness of two strings. Most existing spell correction methods have been introduced and evaluated in the context of speed and precision. Though a lot of these methods are applicable to the above problem, their adequacy has not yet been evaluated. Spelling correction to a specific domain is quite different from natural language spell correction. We propose a method to overcome the isolated non word spelling correction problem in medical domain.

In this paper we evaluate two machine learning techniques on our feature list and then benchmark them to see their effectiveness. In section II we discuss a range of similarity measures including edit distance, dice coefficient, regular expression, 'headers' technique and lig3, which are to be considered for our method. We then go on to introduce a contemporary spell correction method using machine learning techniques in the subsequent section. Thereafter, the two machine learning techniques (integrated in the method) are evaluated. Our goal was also to explore the scope of the two machine learning techniques including linear regression and neural networks using our feature list. Conclusions are drawn in the last section.

II. MEASURES OF SIMILARITIES

Before proceeding in depth into our experiments, it is useful to describe the features that compute the degree of similarity between a pair of words. Each approach uses a unique way to determine the similarity measure for a pair of words. As the best suited approach (for the specific language model/ domain) is unknown before training, we give the privilege of selecting the approach or having a collaborative approach to the model to train itself. We offer this privilege by providing all the similarities measure scores and let the model decide on which approach to choose. This implicitly happens as the model itself generates the proportional mature weights for each similarity score from training and learning.

Our method is flexible and extensible to any number of similarity measures. We have used five different similarity measures including *edit distance, dice coefficient, lig3, regex* and *headers* for our experiments. The headers similarity measure could be further classified more. We have used four such classifications which are further explained along with the other similarity measures in the section.

Minimum number of operations required to transform one word into the other is the *edit distance* value for that instance (pair of words). Insertion, deletion, substitution and transformation are the possible operations with a unit cost. Consider an example to determine the edit distance between strings "kitten" and "sitting".

Step 1: kitten => sitten (substituting 's' for 'k')

Step 2: sitten => sittin (substituting 'i' for 'e')

Step 3: sittin => sitting (insertion of 'g')

Edit distance for this example is 3, as each operation cost one unit. Edit distance and degree of closeness are inversely

proportional.

Similar to n-gram, every sequence of two adjacent elements in a string is a *bigram*. Dice coefficient similarity measure is defined with number of common bigrams and lengths of two strings.

$$Dice\ coefficient = \frac{(\text{number of common bigrams} * 2)}{(l1+l2)}$$

Dice coefficient between two strings "pair" and "hair" is 0.5, which is obtained by substituting values for the parameters i.e. number of common bigrams would be 2 ("ai", "ir") and $l_1 = l_2 = 4$.

Lig3 is simply an extension to Levenshtein Distance. Lig3 score for a pair of words is given by

$$Lig3 = (2 * \frac{(\text{wordlength} - \text{levenshtein distance})}{(2 * \text{wordlength} - \text{levenshtein distance})})$$

We are also introducing *regex* and *headers* to be a measure of similarity in this paper.

Generally, *regular expressions* (abbreviated regex or regexp) are used for pattern matching. The essence behind the approach is that words pertaining to the same language model can be generalized using a specified regex i.e. all English words fall into the pattern [a-zA-Z]*. Based on the assumption, we generated regex for the input words. So, for example if the inputted word was *cntrst* then it's regex could be "[c]\w*[n]\w*[t]\w*[r]\w*[s]\w*[t]\w*". So the intuition is that the correct word *contrast* would have the letters 'c', 'n', 't', 'r', 's' and 't' in the same order as in the incorrect word. Regex value for a pair of words is assigned 1 if regex patterns generated for the two words are a match, else assigned to be 0. Regex generated for an input (a misspelled word) is compared against the regex generated for the words in training set and dictionary to find close matches in the dictionary.

From the study [?Which states that most(>90%) of spelling errors are not with initials(bibliography needed here)], we infer that most of the initials are not misspelled and most of the spelling errors are either due to partially spelling or misspelled. Hence, we included *headers* to check for a match in the first 'n' letters of an input word with the list of words in the dictionary. The algorithm outputs 1 to the instance if the headers are a match and 0 otherwise.

An example is a combination of two strings, in which one could be input string/one entry from training set combination with one entry in dictionary set.

We introduced slight variations to extend the scope of *headers* and further classify the match of "first n words" into *a*) *Header 3*: First three characters of the input string and strings in the dictionary are checked for a match. *b*) *Header 3 without vowels*: First three characters of a string, ignoring the vowels are checked for a match. *c*) *Header 4*: First four characters of the input string and strings in the dictionary are checked for a match. *d*) *Header 4 without vowels:* First four characters ignoring the vowels are taken into consideration for the match.

III. A Machine learning method for isolated spell correction

In this section, we illustrate various phases that are common to both techniques (linear regression and neural networks).

A training set is a set of inputs mapped to their outputs to train the model. The training set contains a pair of input and expected output (one to one mapping). A dictionary is a collection of words in one language model, here preferably English. Training set and dictionary could be specific to one language model. The vocabulary defined for a domain is termed as language model here. Say medical language model contains data from medical domain i.e. words that belong to medical domain.

[Common model diagram to be added here]

A training model is created using the training set and dictionary. Similarity scores including *edit distance, dice coefficient, lig3, regex* and four variations of the *header* algorithm (as discussed in the terminologies section) are calculated for the instances procured from the entries in the training set to the entries in the dictionary. Say there are *t* entries in the training set and *d* entries in the dictionary, hence there would exist *t *d* instances for one training.

Using the similarity scores, depending on the machine learning technique (either linear regression or neural networks here), they model a function F. F is a linear/non-linear function of eight scoring algorithms/ similarity measures. The exact definition of F is different for each technique. For example, F is linear in case of Linear Regression. However it varies for each layer in neural networks.

Both models keep track of their respective error through a cost/error function, though it is different in each case. The *error function/ cost function* is the difference in the accuracy of the actual result to your algorithm's result.

Both models use their own methods for optimization, i.e. error minimization. Objective behind optimizing is to identify the matured weights for the function F so as to minimize the cost/error function from the training.

In both the cases, either NN or LR, once a word is corrected, the incorrect word as well as its correct counterpart is added to the training set to improve the learning at each iteration.

IV. Machine Learning Technique used

A. Neural Networks

In this section we describe a non- linear model using the artificial neural networks for spell correction. Artificial Neural Networks (ANN) are essentially mathematical models which give an output based on a certain set of inputs. It can be described as a non-linear weighted sum of other functions, each having a weight and activation criteria (function)

$$f(x) = K \sum_i w_i g_i(x)$$

Where K = activation function vector

w_i = weights
g_i = functions

A neural network usually has 3 layers – input, hidden and output. Each layer has multiple 'neurons' which take in a weighted, activated input and their output is input for the next layer of neurons.

Algorithm
[flow chart to be added here]

Training Set for Neural Networks

We use the learning capabilities of ANNs to train them for spell correction. As described above, we have many comparison algorithms, each of which gives a similarity 'score' for a pair of words. These scores form the input for our ANN and the output is a cumulative score – a value between 0 and 1. For training, we give similarity scores for a pair of known words (an incorrect word and corresponding correct word in the training set) and the output is 1. We also give some pairs where the incorrect word does not match the correct word with output as 0.

After training, we input an incorrect word paired with all words in the dictionary. The dictionary word for which the output is highest is the correct word.

Details about ANN
Table 1 summarizes the parameters that define the layers in our ANN

[Table1 goes here]

The ANN we created to train the model for spell correction has three layers. Input layer has eight neurons for eight similarity score values for all the instances (each instance being a single combination of each entry in training set with combination with each entry in dictionary). Hidden Layer contains *q* neurons to analyze and learn the mature weights, with bias neuron. The number q was taken as 20 following the guidelines given in the paper [*International Journal of Engineering Trends and Technology, Volume 3 Issue 6, 2012 pg 714 – 717, Approximating Number of Hidden layer neurons in Multiple Hidden Layer BPNN Architecture, Saurabh Karsoliya*] [this is bibliography]

The challenge here was to choose the appropriate transfer functions and the suitable number of neurons for each layer.

Activation Functions:

1. We wanted the outputs to be a directly related to the inputs (i.e. the similarity measures); we did not define any transfer function to the first layer.
2. For second layer, we needed a sigmoid type function for its asymptotic behavior. We choose hyperbolic tangent function as this gave the best results, possibly due to the negative outputs giving greater flexibility to the network model
3. For output layer, as we wanted to restrict output values to a range of 0-1, hence we choose the basic sigmoid function

The first layer contains (*m*+1) neurons in case if there were *m* similarity measures considered for an instance. Each neuron to input each similarity measure and the last neuron is the bias. Bias gives greater flexibility to the model, it allows for a constant value in the network functions. Some functions cannot be modeled without bias [stackoverflow reference].

Error Calculation and Training

For calculating error we chose was *Mean Square Error (MSE)* method. Due to the sigmoid activation function in the output layer, the output was between 0 and 1. Our expected outputs were either 1 (if match) or 0. The error for each data point is also between 0 and 1. The Back Propagation algorithm was used for training the weights.

B. Linear Regression

We have two reasons to choose linear regression. One, shortage of training data: it is generally acknowledged that a high bias (underfitting) model is more suitable than a high variance model in cases where the training data is less. This is because a high variance model contains a high proportion of parameters relative to the number of observations, which can lead to overfitting i.e. a model is not likely to generalize the trend and perform well in a new test data set. Two, data in medical domain is variable - We found out that in the medical domain, spelling errors have a large variance i.e. we are likely to get huge variations in the complex model function F when trained with different training sets. In order to generate lesser variations, we intended to create a more generic linear model.

The Error of a model can be expressed as

E[C] = Variance(f(x)) + [Bias(f(x))]2 + variance(e) Variance(e) is an irreducible error and it does not depend on the training set to be reduced.

We had to make a tradeoff between variance and bias, based on the reasoning given above we found out that a model with high variance is more likely to cause more error than a model with high bias i.e. a linear regression model.

In this section, we adopt linear regression to be the machine learning technique and gradient descent as the optimization method to determine the matured weights.

The function F modeled in this linear model is the weighted sum of similarity measures.

Y=n1x1+n2x2+...+n8x8 algo1

where x is vector of i similarity measures. In the case of spell correction, similarity measures represent the scores given by each 8 algorithms explained above and n1, n2, n3,…, n8 are the weights to be obtained by the training model.

Details about Linear Regression Implementation
We can measure the closeness of any 2 words by calculating the cumulative similarity score given by algorithm [algo1]. To correct an incorrect word, given a master dictionary, we can

measure the closeness of that word with each word in the dictionary. The best match will be the dictionary word with the maximum cumulative similarity score.

Error Calculation and Training

To get the above algorithm working, we need values for the individual weights n_1, n_2, etc. We obtain these weights using linear regression. We use a training set, having a set of incorrect and correct words. For cost function, we use rank of the correct word, on sorting all dictionary words.

We use gradient descent to minimize the cost function.

For every incorrect word in the training set of size *t*, we compute each feature (x_1, x_2, x_3,..., x_8) for all words in the dictionary and rank them according to the normalized similarity measure values. So, if the size of the dictionary is *d*, then there will be *t*d* records, for which we compute *k* similarity measures for every such instance.

Our cost/error function approach being,

$$C = (F_{maxword} - F_{correctword}).$$

The key idea is to identify and rank the difference in the position of the training set correct word in the dictionary with the word that got rank as 1. The intuition is that the correct word corresponding to the incorrect word in the training set should have the rank 1 (in a mathematical model, typically C should be equal to 0 in this case). As in this case, error, being an indispensable part of our function, C is expected to be minimum or negligible. The threshold for the error is decided by the experimenter, which is set according to the precision expected.

Different statistical estimation methods are available to minimize the error. The perfect set of weights n_1, n_2, .., n_8 (unknowns) of the function F, that minimizes the Function C are estimated using this procedure.

V. Results

a) Through this paper we would like to stress and point out that Edit Distance measures tend to be very close to 1 or 2 for most spelling corrections in the Natural Language but not so in the case of domain specific words. There were many words in our training set for which the Edit Distance was easily 4 or even 5. The results below show that in case of domain specific words (like in the medical domain) the Edit Distance can be fairly high and we would have to consider such high distances. The graph below, shows that even at high distances, the accuracy and precision do not fall significantly.

[precision comparison graph to be added here]

b) Comparison of the two techniques (linear regression and Neural Networks) for isolated word spell correction is shown below.

[Table to be added here]

c) Contribution of each feature

[Graph should be added here]

d) Hence, from the work carried we understand that our algorithm is good for natural language model spelling correction. We also infer that it is better for domain specific model, like medical terminology.

VI. Acknowledgement

VII. References

List and number all bibliographical references in 9-point Times, single-spaced, at the end of your paper. When referenced in the text, enclose the citation number in square brackets, for example: [1]. Where appropriate, include the name(s) of editors of referenced books. The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in "[3]"—do not use "Ref. [3]" or "reference [3]". Do not use reference citations as nouns of a sentence (e.g., not: "as the writer explains in [1]").

Unless there are six authors or more give all authors' names and do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, "Title of paper if known," unpublished.

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989