CS 229 project report
Younggon Kim (younggon@stanford.edu)

Smarter text input system for mobile phone

1. Abstract

Machine learning algorithm was applied to text input system called T9. Support vector machine (SVM) is used to predict the word. The experiment shows the algorithm can reduces error by 40-50%.

2. Background

2.1. T9 system

26 alphabets are mapped to 8 numeric keys (from 2 to 9). A user presses one key for each alphabetic letter. For example, "ACE" is entered by pressing "223". Because mapping is many-to-one, converting sequence of number to alphabetic word can be ambiguous. A conventional phone picks most common word among the valid words in a dictionary.

If the word displayed is not the user want, the user press 0 to display other word until the right word appears. If word is right, the user presses # to enter a space and moves on.

2.2. Room for improvement

Conventional algorithm is context insensitive. In other words, it does not look at other words to predict. If context is used for decision, the accuracy of prediction might be improved.

3. Framework

3.1. Notation

Here are several terminology used throughout the paper.

token : A word - sequence of alphabet letters. Token does not contain white space.

encode(): A function which maps token to code, as described in 2.1.

code: A sequence of numbers mapped from a token.

candidates(): A function which maps code to possible tokens,

I.e. candidates(code) = {token | encode(token) = code}

3.2. Problem formulation

Using these definitions, the problem can be formalized as following:

Input: sequence of tokens followed by code

$token_k$, $token_{k-1}$, … $token_2$, $token_1$, code

Problem: choose right token among candidates(code)

Note that there is only one code in the input. In T9 system, a user gives feedback for each word entered. Therefore, the system knows the right tokens for all previously entered codes.

3.3. Input text

It is good to have an enough input text to train and verify. I decided to use free text files available in http://www.gutenberg.org/. The site has a number of books which does not have copyright problems. For example, it has 'the adventure of sherlock holmes by Conan Doyle' and 'the war of the worlds by H. G. Wells'.

The novels do not represent well texts used in messaging. They are more formal and tend to have long sentence. For the sake of getting the data, this project confined the scope to written language.

4. Algorithms

4.1. Simple word counting

This algorithm is used to emulate the behavior of conventional phone. Additionally, this algorithm serves as a reference to evaluate proposed algorithm.

Word counting algorithm is simple. It keeps counts for each word appearing in the training examples. Prediction is made by comparing the counts among candidates(code). A token which has highest counts is chosen.

4.2. SVM

4.3. Binary classification

Let's begin discussion with a binary classification. The algorithm will be extended to multiclass case in the later section.

In a binary case, number of candidates is two. For example, code 63 has two candidates - 'of' and 'me'. Every sentence includes 'of' and 'me' is a training example for code 63.

Let's say that the sentence is "give me a box of chocolate". This sentence has two training examples for code 63.

input sentence: give me a box of chocolate

example 1 for code 63: give

example 2 for code 63: give me a box

Example 1 is classified as 'me' and labeled as '+1' while example 2 is classified as 'of' and labeled as '-1'.

4.4. Feature vector

Each example need to be converted to feature vector for processing. For the sake of simplicity, assume total number of tokens is six.

token0: a

token1: box

token2: chocolate

token3: give

token4: me

token5: of

Feature vector indicates whether example has the token or not. Using above examples,

vector for example1: [0 0 0 1 0 0]

vector for example1: [1 1 0 1 1 0]

4.5. Encoding a distance

Above scheme does not take the distance into account. In example 1, 'give' is right before the code while 'give' in example 2 is four tokens away from the code. Intuitively, the correlation will decrease as the distance gets longer. To encode this knowledge, exponential function $2^{-distance}$ is used instead of '1'. Following is a result from this encoding,

vector for example1: [0 0 0 1 0 0]

vector for example1: [2^{-1} 1 0 2^{-3} 2^{-2} 0]

4.6. Evaluation

Once SVM algorithm learns from examples, each code has its own w and b parameters. To predict token, calculate w * (feature vector of test case)' + b.

4.7. Extending to multiclass

Binary classification algorithm can be run several times to support multiclass. For example, code 6553 has three candidates – mind, mine and nine. SVM algorithms are run three times with different classification labels. In the first run, examples for token 'mind' are labeled as '+1' and others are labeled as '-1'. Similar steps are taken for second and third run.

After algorithm learns, each candidate has w and b. Evaluation requires several calculations, i.e. calculate w * (feature vector of test case)' + b for each candidate. Choose the token which has the largest value.

5. Implementation

Script language Ruby is chosen because of its flexibility and string, hash ADT. Initial implementation is too slow to run on a large dataset. A few optimizations are made – some is a trade off between accuracy and speed.

5.1. Sparse vector

Feature vector is sparse. Instead of using basic array, customized ADT is built. Dot product between feature vector became much faster. It sped up the algorithm considerably, because each learn requires m*m/2 inner products, where m is the number of training examples.

5.2. Limiting distance

If token is far away from the code, it does not contribute much to predict. Max distance is limited to MAX_DISTANCE which is adjustable parameter. Any token has larger distance than MAX_DISTANCE is ignored when building feature vector.

Several experiment had been conducted by varying MAX_DISTANCE. 7 was chosen because going above 7 marginally increased accuracy.

5.3. Limiting number of training examples

Number of training example has direct impact on running time. Some common word such as 'of' has a lot of training examples. To set upper bound of running time, maximum number of training example is limited to MAX_NUM_EXAMPLES. Each candidate can not have more than MAX_NUM_EXAMPLES examples. If any candidate has more than this, extra examples are randomly dropped.

For example, if MAX_NUM_EXAMPLES = 400, maximum training examples for code '63' is 800 which is composed of 400 for token 'of' and 400 for token 'me'

MAX_NUM_EXAMPLES had a direct impact on accuracy as well. More on this will be

discussed in results section.

6. Training and Evaluation

70% of sentence is used for training and 30% is used for evaluation. Two different modes of experiment were conducted. In the first experiment, only one book, 'Adventures of Sherlock holmes', is used. In the second experiment, eight books are used. The input to the second experiment should have more variety on the selection of words and style of writing.

In the evaluation phase, each word in the sentence is predicted by two algorithms. There were several cases.

Trivial: only one candidate exists. No prediction is necessary

Out of dictionary: the algorithm has never seen the token in the training set. Both algorithms can not predict correctly.

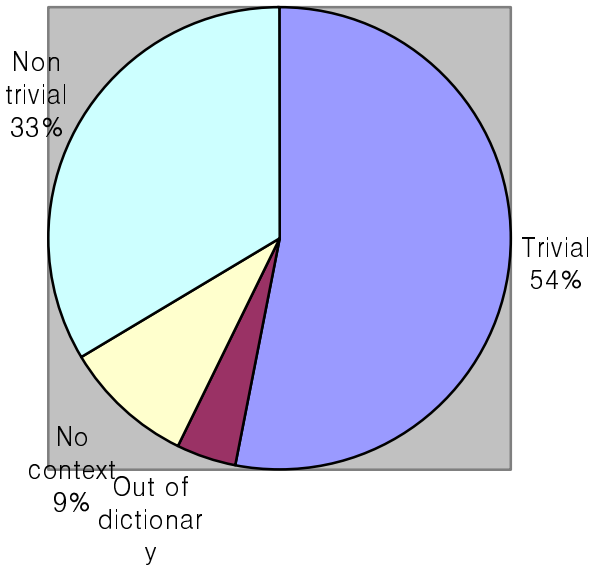
No context: test case does not have enough contexts to use SVM. There are two cases. One case is when the word is at the beginning of sentence. The other case is when test vector has no overlap with 'any' training examples. In other words, w * (feature vector of test case)' becomes zero for all candidates.

Non trivial: all other cases in which both algorithms competes.

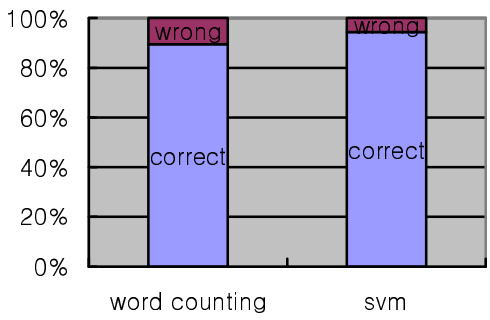
7. Results

7.1. experiment 1 – one book with 30,000 words

word distribution

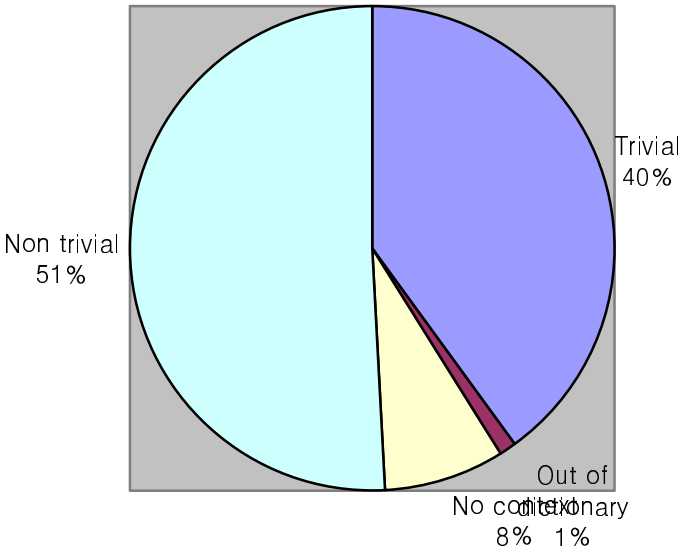


prediction result



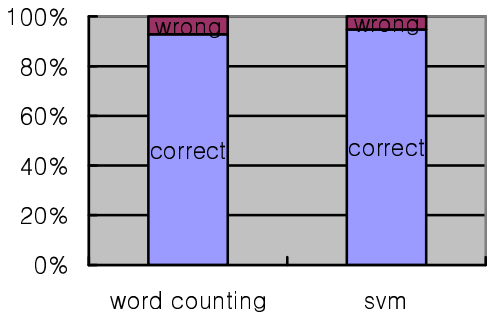
7.2. experiment 2 – eight books with 396,000 words

word distribution



\

prediction result



7.3. comparison

In experiment 1, SVM's number of error is 53% of word counting's. In experiment 2, SVM's

number of error is 73% of word counting's. SVM outperformed word counting by factor of 2 when one book is used as input. This advantage is reduced when eight books are used as input. This is related to the fact that same MAX_NUM_EXAMPLES of 800 is used for both experiments. Second experiment has input data set more than 10 times larger than the data of first experiment. To achieve similar performance, MAX_NUM_EXAMPLES should have been increased as well. Due to the large running time, however, MAX_NUM_EXAMPLES was kept same.

8. Conclusion

SVM is implemented to improve accuracy of T9 input system. The error was reduced by 50% in the best case. However, long running time of the algorithm prevented it to maintain same performance in a large dataset.