Winnow Based Grammar Correction

Adrian Moore

UW Madison Computer Sciences
302 South Park Street
Madison, WI, 53706
Stewsters@gmail.com

Abstract

I attempt to perform context-sensitive grammar correction on frequently confused pairs of words. This provides a machine learning problem with a high dimensional space where many factors are irrelevant. I implement a system similar to Winnow-2 to allow for generating a decision hyper-plane while still allowing fast online training. I compare the weights from two instances of Winnow to determine the correct answer.

The Algorithm

The algorithm I use is based on two instances of winnow-2 running in parallel for the word given and the matching pair word.

First we define m pairs of confused words P.
P = (Pα1,Pβ1), (Pα2,Pβ2), … , (Pαm,Pβm)
d = distance from a word to search = 10
Θ = target weight = D * 2 = 20
α = weight update = 2
D = document, array of words.
w = weight for a word. Separate dictionaries for each word in a pair, with an initial value of 1 for each element.

Classification:

If the summation of the weights of the nearby words exceeds the summation of the weights of the nearby words for the opposite word in the pair, then the use is classified as correct. Otherwise it is incorrect.

$\Theta = \sum_{k=i-d}^{i+d} w\left[D[k]\right] - \sum_{k=i-d}^{i+d} w[P[D[k]]]$
If Θ > 0 then the example D[i] is correct.
Otherwise D[i] is incorrect.

Update:

If an example classifies correctly, our weights work well and do not need to change.

If an example is classified as correct, but was not, then for all weights involved, w = w / α

If an examples is classified as incorrect, but was correct, then for all weights involved, w = w * α

This algorithm runs in O(N*p*d) time, where n is size of the document, p is the number of pairs to look for, and d is the width of the search. I hold d and p constant, so it runs in linear time with respect to the size of the document.

To train the data I run update on a set of positive examples, and then flip the paired word and run it on the negative examples for the other paired word.

Implementation

My implantation is composed of both a command line grammar checker and a plugin for GEdit.

To increase the amount of data my algorithm had to work with I stripped out all punctuation. This causes the following pairs to not work: (your, you're) (their, they're) (i.e., e.g.) (its, it's). Instead I can phrase them like (your, youre). Note that this means that I still can't use (its,its). Some pairs that I originally searched for (climactic, climatic) do not exist in the corpus, so I removed them.

Results

The training corpus I used was a set of 15 ebooks from Project Gutenberg and the Cyberpunk Project, using 5 books from one author, and the rest from variety of sources. If the nearby words had less than 2 previous examples, it classifies it with an additional notEnoughInfo tag. If one word is only seen once near a pair, it gives us little information. Due to the sparseness of my data, these make a significant portion of the training.

On a related set of text, with the same author and genre as about 1/3 of the training corpus and with online training turned off:

Positives- Correct Words
650/676 correctly classified with enough info
96.15% Accuracy
2294/2771 correctly classified without enough info
82.78% Accuracy

Negatives – Flipped Words
644 / 675 correctly classified without enough info

95.41% Accuracy
2331 / 2752 correctly classified without enough info
81.07% Accuracy

I believe this is over-training, because an author will typically use similar words in his works, especially if I include the two sequels to the book in my corpus. However, if this algorithm is used for long enough for a single person's grammar correction, it would be beneficial if it adapted to their style of writing.

I also tested an unrelated piece of text from a different genre with a different author (a medical textbook).

Positives – Correct Words
703 / 779 correctly classified with enough info
90.24% correct
6304 / 8239 correctly classified without enough info
76.51 % correct

Negatives - Flipped Words
679 / 779 correctly classified with enough info
87.16% Accuracy
6176 / 8239 correctly classified without enough info
74.96% Accuracy

We can see that this works significantly less well. This is partly because a medical textbook uses a lot of words that are not seen in popular literature. You can see that the ratio between enough and not enough info is much worse. It is therefore important to train a model using as large a corpus as possible, with at least one text that relate to the topic.

Future Work

The easiest way to increase the accuracy of algorithm is to give it more data, which would mitigate the problem of data sparseness. If online learning is re-enabled, the accuracies are far better. This is because it trains on the same document that it is correcting, so language usage is closer than any other example I could provide. This is intense overtraining, but during the normal operation of a grammar checker with a human running it in a program, this data would be available.

Additional features that should be considered would be to find bigrams and trigrams that include one of the pairs within the text. If the bigram "eat desert" was seen, it is likely it should be corrected to be "eat dessert". Unfortunately, this requires a lot more text to train on to automatically find these n-grams, something that I did not possess. Alternatively, I could have manually defined these relations, but it would have required a lot of work and would not adapt well to new data or language patterns.

Another feature that could be considered is reuse of patterns. If "run too fast" is seen 10 times in a document, it is possible that "run to fast" is a typo.

I strip punctuation to get enough similar data. Using a good stemming algorithm instead might produce better results. Also using a part of speech tagger and then adding the tags to the feature set should boost performance. Some words are only seen in certain parts of speech, where their pair might be used in a different part of speech.

For words of up to distance 10 I count as equal weighted when training. The further away a word is from the word we are checking, the less likely it is to have the same context. If I change the amount that the weight changes when training with respect to the distance from the checked word, it can be made to account for this.

References

Golding, A., and Roth, D. eds. 1999. *A Winnow Based Approach to Context-Sensitive Spelling Correction.* Machine Learning.: Springer Netherlands.