Count-based Spelling and Grammar Correction

1 Problem Statement

Spelling and grammar errors infest the lives of everyone from native to non-native speakers of a language. We will consider the problem of spelling and grammar correction. Traditionally, spellchecking utilizes a dictionary, but this creates the problems of out-of-vocabulary and data sparseness. To overcome this many have used a corpus, a database of existing texts, usually hundreds of books, and newspapers. Corpora can be tagged or untagged. A tagged corpus contains, for each word, a tag describing which part of speech the word represents. The corpus is usually preprocessed into counts of n-grams. An n-gram is a sequence of tokens, usually words or characters of length n. Different approaches to solving the problem include using tagged or untagged corpora and using n-grams with different n in different parts of the error correction process.

In a related field, machine translation does not traditionally utilize much language data (except for the translation model), which [4] suggested could improve translation. This suggests a naive improvement for machine translation: run a spell checker on poorly translated text.

We propose using an untagged corpus and counts of combinations of different sized n-grams to check for spelling and grammar mistakes in texts and translations, and offer corrections based on these counts.

2 Related Work

Traditional techniques include using a language model including a grammar model of the language. Relatively recently ([3] published one of the first datasets in 2006), other techniques using probabilistic models of language with much bigger datasets have started taking over and offered much better accuracy.

The technique described by [2] uses 3-grams consisting of words and parts-of-speech and checks for errors based on their probability of occurrence in a tagged corpus. This method is insufficient because it doesnt vary the size of the n-grams. It also incorporates part-of-speech, which increases permutations and complexity.

The algorithm by [1] is comprised of a spelling error checker using unigrams information from Yahoo! N-Grams Dataset; a candidates generator based on a letter-based 2-gram model; and a context-sensitive error corrector that selects the best spelling candidate for correction using 5-grams information from Yahoo! N-Grams Dataset.

3 Implementation Details

Our implementation is based on known approaches utilizing three phases:

The first problem is error detection, in which we will utilize 3-, 4- and 5-gram counts to score each word in a sentence, instead of using probabilities of 1-grams as [1] does, or part-of-speech tagging as [2] does. This makes sense because an uncommon sentence that exists in the corpus will show up as correct in our algorithm, while other techniques will flag it as probably wrong. This would also give better feedback to users about how common their phrasing style is, which traditional grammar checkers do not do. We also do not require a tagged corpus, which is hard to make, and which makes the algorithm more complex. We will do this using different combinations of 3-, 4- and 5-grams to compare effectiveness.

The second problem is generation of a confusion set, the possible corrections. We will do this using a simple brute-force search in 1-grams for correct words. We will also generate permutations of words to correct for word order errors.

The third problem is choosing a correction, which we will do by ordering the confusion set by the score given to the corrected sentence. The score is calculated as above.

4 Results Evaluation

In order to optimize our algorithm we will compare different versions of it using n-grams with different ranges of n. We will compare it against the approaches by [2] and [1] and to existing spell checkers. We will compare percentage of errors successfully corrected, errors not corrected and errors falsely corrected. As discussed earlier, we improve upon the approach of [2] by using an untagged and bigger corpus and upon [1] by using a score based detection algorithm. This should give our algorithm a better detection rate than either of the others. Our algorithm will also do a more extensive search for corrections, trying to correct more severe errors that arent possible by the other algorithms.

References

[1] Bassil, Y. Parallel spell-checking algorithm based on yahoo! n-grams dataset. *Science 3*, 1 (2012).

[2] Fossati, D., and Eugenio, B. D. A mixed trigrams approach for context sensitive spell checking. In *Computational Linguistics and Intelligent Text Processing* (2007), vol. 4394, pp. 623–633.

[3] Michel, J., Shen, Y., Aiden, A., Veres, A., Gray, M., Pickett, J., Hoiberg, D., Clancy, D., Norvig, P., Orwant, J., et al. Quantitative analysis of culture using millions of digitized books. *science 331*, 6014 (2011), 176–182.

[4] Norvig, P. Natural language corpus data. *Beautiful Data* (2009), 219–242.