Investigation into Grading English Grammar

Martha Bellows Antoinette Bongiorno Brandon DiGiulio

May 8, 2017

1 Abstract

Summary of paper.

2 Introduction

Discuss previous paper and how we feel native English speakers grading sentences is not a good metric (reproducibility, subjectivity, credentials, etc.). How we aim to solve this issue in our paper and brief overview of our process.

2.1 Motivation

"We invited 3 English native speakers to do this evaluation." – original paper

- Subjective
- Unreplicable
- Inconsistent

We recognize that there is a need for a more objective method of scoring the grammaticality of sentences in order to further research into the growing field of Automatic Text Summarization. Grammaticality evaluation will help rank competing systems as well as provide developers of new solutions with a means to measure the effectiveness of their systems.

Some examples of citing formatting.

Many hospitals face significant budgetary concerns and, due to Medicare regulations, excessive readmission rates reduce Medicare payments. In the United States, readmissions cost hospitals an average of 17.4 billion dollars per year [1]. These attempts have varied in success with results averaging around 60 percent accuracy by using raw data [2].

2.2 NLP Community

- What is the community standard?
  - ROUGE metric
  - Human Judgement
- How is human judgement used?
  - Subjective scoring on 1-5 scale
  - grammaticality, non-redundancy, clarity, coherence
  - Often random participants, native English speakers
  - DUC conference uses a set of questions for assessors
- Why hasn't a more objective method been used before?
  - There are commercial solutions to grammar checking
  - Aside from that, the NLP community has not adopted a more objective method
  - Very few instances of previous research, Quantitative evaluation of grammaticality of summaries (Ravikiran Vadlapudi and Rahul Katragadda, 2010)

2.3 Grammar

What is grammar?
The system of rules and syntax that defines

how things should be written, spoken. Gives communication an understood, defined meaning between two or more parties.

3 Parts of Speech Tagging

Our idea is to develop a way to score the grammaticality of an input sentence based on the comparison of that sentence?s POS tag sequence to a generated grammar rule set. This grammaticality scoring method could someday replace human judgement in Automatic Text Summarization evaluation.

3.1 Natural Language Toolkit

In order to do the work of tagging sentences with parts of speech (POS) tags, we utilized the Natural Language Toolkit (NLTK). [3] NLTK is written in python and has many functions for use with natural language processing. In particular, we used NLTK in order to tokenize and tag sentences to build a text file of POS tag sequences. The tags used by NLTK is the Penn Treebank Tagset.

Tag	Description
$	Dollar sign
"	Quotation mark
(	Opening parenthesis
)	Closing parenthesis
,	Comma
–	Dash
.	Sentence terminator
:	Colon or ellipsis
CC	Conjunction, coordinating
CD	Numeral, cardinal
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or conjunction, subordinating
JJ	Adjective or numeral, ordinal
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal auxiliary
NN	Noun, common, singular or mass
NNS	Noun, plural
NNP	Noun, proper, singular
NNPS	Noun, proper, plural
PDT	Pre-determiner
POS	Genitive marker
PRP	Pronoun, personal
PRP$	Pronoun, possessive
RB	Adverb
RBR	Adverb, comparative
RBS	adverb , superlative
RP	Particle
SYM	Symbol
TO	"To" as preposition or infinitive marker
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, present participle or gerund
VBN	Verb, past participle
VBP	Verb, present tense, not 3rd person singular
VBZ	Verb, present tense, 3rd person singular
WDT	WH-determiner
WP	WH-pronoun
WP$	WH-pronoun, possessive
WRB	WH-adverb

Figure 1: Penn Treebank Tagset

Other systems were considered such as RASP (Robust Accurate Statistical Parsing) which is another system designed for syntactic annotation of free text. [4] It is implemented using C and Common Lisp and uses POS tags derived from the CLAWS tagset. The CLAWS tagset is comprised of over 130 tags depending on the version so it is much more extensive than the Penn Treebank Tagset. For example, the CLAWS tagset goes into finer detail with verbs spec-

ifying individual tags for the verbs to be, to do, and to have. It even goes so far as to break out each of these verbs to have a tag per verb tense. Because RASP uses a larger tagset, we deemed it to be beneficial to start with the simplest tagset available.

Another system we considered is the NLP software available from The Stanford NLP group. Like NLTK, it uses the Penn Treebank tagset but instead of being implemented in python, they use Java. In addition, the documentation available from NLTK is better which is why we chose NLTK over other systems.

3.2 NLTK - Tagging Sentences

In using NLTK, we are able to transform regular sentences into POS tags. There are four stages in this process from getting it from the initial sentence to a sentence with POS tags. Starting with initial sentences (1), it is necessary to convert this so each sentence is broken down to its individual sentences (2). This process is known as tokenizing sentences. The next step is to break each sentence into its individual words and punctuation which is called word tokenization (3). The final step is to tag each word in the sentence (4).

1. Initial Sentence: 'I went on a walk. It was nice outside.'
2. Tokenize Sentence: ['I went on a walk.', 'It was nice outside.']
3. Tokenize Words: [['I','went','on','a','walk','.'], ['It','was','nice','outside','.']]
4. Tag Words in Sentence: [('I', 'PRP'), ('went', 'VBD'), ('on', 'IN'), ('a', 'DT'), ('walk', 'NN'), ('.', '.')]

3.3 Project Gutenberg

In order to build the rule set of correct tag sequences, literature from Project Gutenberg was used. [5] Ten famous literature works were used. They were picked based on their popularity and the fact that the authors are all native English speakers. Because of their native English speaker and their world-wide renown, proper grammar is assumed. The ten pieces we used are as follows.

1. *The Adventures of Tom Sawyer* by Mark Twain
2. *Alice in Wonderland* by Lewis Carroll
3. *Emma* by Jane Austen
4. *Jane Eyre* by Charlotte Bronte
5. *Moby Dick* by Herman Melville
6. *The Picture of Dorian Gray* by Oscar Wilde
7. *Pride and Prejudice* by Jane Austen
8. *Sense and Sensibility* by Jane Austen
9. *A Tale of Two Cities* by Charles Dickens
10. *The Yellow Wallpaper* by Charlotte Perkins Gilman

4 Algorithm

4.1 Generating Rule Set

In this section, we go over how our algorithm generates its underlying rule set. Based on existing, assumed grammatically correct, pieces of work [5], we build a pseudo-database of correctly formed grammar combinations. Books are first read into memory by the application, one at a time. Using the Python NLTK library, each book is then tokenized into separate sentences and then further tokenized into individual words per sentence. Each sentence is then treated separately, as each word contained within is tagged with the appropriate Part of Speech (POS) tag generated by the NLTK library. The result, is a list of tuples, each containing a word of the sentence and the POS tag. Since our algorithm will only care about the POS tags to check grammar, the original word is removed and all that remains is the sequence of POS tags for each sentence.

Next the algorithm compares these new sequences with the current database to check for any existing matches. A binary search is done (the database is kept sorted) and if a match occurs, it will not be inserted as to avoid duplication of sequences. If there are no matches found at the end of the binary search, the sequence is added and the list sorted. Through this part of the overall algorithm, a ruleset to compare summaries and other sentences to has been created.

Algorithm 1 - Generate Rule Set

Inputs:
Sentence or sentences known to be grammatically correct.

Outputs:
Sorted .txt file containing all sequences of the sentences' POS tags.

Method:
1) readFile("...")
2) tokenizeSentences
3) convertToTags():
for sentence **in** allSentences:
- Tokenize each word of the sentence
- Tag each word with a part of speech
- Remove word from tuple, just leaving POS tag
- Add POS sequence to list

end for
Return the list of POS sequences
4) writeTagsToFile():
for sequence **in** listOfTagSequences:
if sequence is **not** already **in file:**
- Add sequence to file and then sort

end if
end for

Figure 2: Algorithm 1 - Generating Rule Set

4.2 Grading Against Rule Set

In order to grade now against the aforementioned ruleset, we must now compare the grammar sequences of sentences to be graded with the ones that we already know to be correct. The new sentences must first be handled as the ones were in Section 4.1. They are first read and then processed using the NLTK library. Again, this tokenizes them into individual sentences, then words, and finally broken into sequences of POS tags.

It will next loop through each sequence generated from summary or set of sentences and search for a matching sequence in the established ruleset. A binary search is per-

formed on the ruleset and if a match is found, the sentence will receive a score of '1'. If a sentence does not find a matching sequence, it is assumed grammatically incorrect and thus will receive a score of '0.' Summing up all the scores across the input sentences, it is then divided by the total number of sentences to get an average score. This final score is the given grade for the summary?s grammatical correctness.

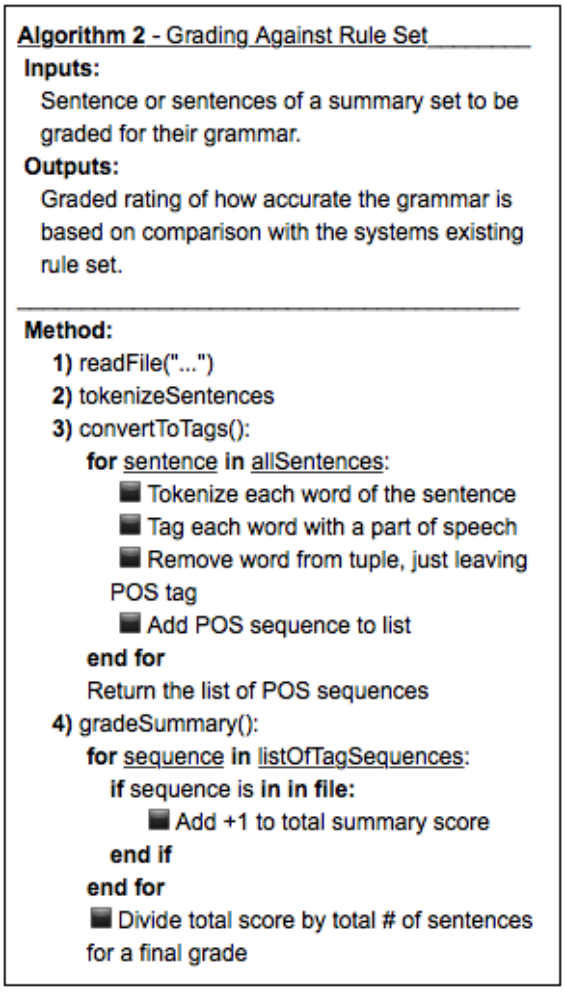

Figure 3: Algorithm 2 - Grading Against Rule Set

5 Results

We expect that through grading summaries, grammatically correct sentences will have resulted with a score of '1' and grammatically incorrect sentences scored with a '0'. As the ruleset grows with the addition of more books, we also expect to see less false-negatives.

The ruleset was initially established with the input of ten classical books from Project Gutenberg resulting in 56,025 unique POS tag sequences to be used for grammatical comparison and grading. Preliminary results showed that it would find and score correctly sentence structures that were known to be within the ruleset. However, the ruleset was not robust enough to cover everything and thus there were some known grammatically correct sentences that were still being given a score of '0.'

The algorithm was fed a control of one grammatically correct paragraph from one of the books included in our ruleset, one grammatically correct paragraph from an auto-summarizer, and one modified paragraph that was known to have a few grammatical mistakes.

6 Limitations

When creating the text file of correct POS tag sequences, it became apparent that it would continue to grow significantly no matter how many sentences were already in-

cluded in the file. When added the ten books to it, it was apparent that there is not much overlap in sentence structure between each book. For example, the total number of sentences in all ten books is 58,895 while the tag sequences file has 56,025 sentences. This is a difference of only 2,870 meaning that between ten novels, there is only an overlap of about 5%. This result is a clear example of the infiniteness of language. When discussing sentence construction in the NLTK documentation [3], they comment on how easy it is to "extend sentences indefinitely." They go on to say "it's not hard to concoct an entirely novel sentence, one that has probably never been used before in the history of the language." Because of the infiniteness of language, grammatically correct sentences that are not in the list of grammatically correct sequences is entirely possible.

Another limitation that will start happening with this method as more sentences are added is the text file will continue to grow. As mentioned, the number of tag sequences currently in the file is about 56,000 and that is only after utilizes ten works of literature. As new works are added and the file grows, there will be an increasing need to optimize the algorithms for better storage and more efficient searching.

Limitations on books used

7 Conclusion

Conclusion

8 Future Work

Future work

References

[1] A. Catlin, C. Cowan, M. Hartman, and S. Heffler, "National health spending in 2006: A year of change for prescription drugs," *Health Affairs*, vol. 27, no. 1, pp. 14–29, 2008.

[2] D. Kansagara *et al.*, "Risk prediction models for hospital readmission: A systematic review," *Journal of the American Medical Association*, vol. 306, no. 15, pp. 1688–1698, 2011.

[3] S. Bird, E. Klein, and E. Loper, "Natural language processing with python," 2009.

[4] T. Briscoe, J. Carroll, and R. Watson, "The second release of the rasp system," *Proceedins of the COLING/ACL 2006 Interactive Presentation Sessions*, pp. 77–80, 2006.

[5] P. Gutenberg, Retrieved May 1, 2017, from www.gutenberg.org.