Investigation into Grading English Grammar

Martha Bellows Antoinette Bongiorno Brandon DiGiulio

May 11, 2017

1 Abstract

Summary of paper.

2 Introduction

In the field of Automatic Text Summarization, there are two main categories of research: extractive and abstractive summarization. Extractive summarization describes the process of selecting contextually important sentences from the document and concatenating them to form a summary, while abstractive summarization aims to generate novel sentences that represent the main idea in a more concise manner. With advances in natural language processing and computer science, abstractive summarization and its newer counterpart, compressive summarization, have grown in popularity. Despite this growth in the field, it seems the methods used to evaluate progress have not scaled well, causing the community to continue to rely on human subjectivity as a metric in grading grammaticality and linguistics. In this paper we propose a model for evaluating grammaticality in a more objective and replicable way.

2.1 Motivation

In recent years, there has been research into using Integer Linear Programming for compressive summarization. In a paper by Li, Liu, Liu, Zhao, and Weng, we see the author?s claim to make significant improvements in the linguistic quality of generated sentences using ILP and constituent parse trees[add citation to original paper]. Curious to realize just how significant the improvements were, we looked into their method of measurement. In the paper, Li et al. cites that they "invited [three] native English speakers to do the evaluation" [cite original paper]. Taking note of the lack of replicability of this method, we recognized that there is a need for a more objective way to score the grammaticality and linguistic quality of sentences in order to further research in the growing field of Automatic Text Summarization. A more consistent method of grammaticality evaluation will help researchers measure the effectiveness of their systems, ranking them against competing research.

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

As research began with extractive summarization, it seems that the focus of Automatic Text Summarization is still on content and salience. In extractive summarization, the sentences that comprise the summary have not been altered in a way that could change their grammar, and assuming that the source document has correct grammar, there isn?t a need to score grammaticality. Due to this, the community standard for grading automatic summarization is the ROUGE metric [Yao]. This metric uses a comparison of n-grams between the summary to be evaluated and one or several human-written reference summaries, essentially scoring the content of the summary. Yao, Wan, and Xiao write that aside from content, evaluation on other aspects of summaries relies heavily on human judgement. At DUC and TAC, the main conferences in the field, human judges are asked to rate different aspects of the summaries. This manual evaluation, on grammaticality, non-redundancy, clarity, and coherence, is said to be indispensable as there is no better solution at this time[Yao].

This method of evaluation is used across the community. We see Li et al. asking a small sample of English native speakers to give a rating, on a scale from 1, bad, to 5, good, for both grammar and coherence. Woodsend and Lapata conduct their human judgement evaluations over the internet using Mechanical Turk, using a larger sample, of 54 volunteers. These self-reported native English speakers grade each summary for grammaticality and informativeness, on a scale of 1 to 5, after successful completion of comprehension questions[cite Woodsend and Lapata]. In our research, we are curious as to why a better scaling solution has not been developed, or been applied, before. At this time, there are commercial solutions that grade grammar, but none have seem to have been applied to this community. In the model presented in this paper, we attempt to find a way to bridge this gap, so that automatic summarization research does not have to rely on the subjectivity of human judgement.

2.3 Grammar

In order to grade a summary on grammaticality, a definition for grammar needs to be established. Grammar is, simply put, a system of rules and syntax that defines how things should be written and spoken. Grammar gives communication an understood, defined meaning between two or more parties. These rules can be further defined using words? parts of speech and their relation to each other. "We say a sentence is grammatically well formed when the adjacent words are in agreement with each other. That is, the parts-of-speech (POS) tags of adjacent words are mutually compatible, where the level of compatibility accounts to the degree of acceptance." [3] Using this, we can determine the importance of compatible combinations of POS tags in grading overall grammar.

3 Parts of Speech Tagging

Our idea is to develop a way to score the grammatical correctness of an input sentence based on the comparison of that sentence?s POS tag sequence to a generated grammar rule set. This ruleset can be produced by processing a large number of grammatically correct sentences and analyzing these correct combinations of respective POS tags. By similarly processing automated text summaries into series of POS tags, we have the foundation for comparing a known set of correct sequences to these new, grammatically ungraded sentences. This grammaticality scoring method could someday replace human judgement in Automatic Text Summarization evaluation which will make this grading process more objective, reliable, and repeatable.

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

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). [4] 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.

Other systems were considered such as RASP (Robust Accurate Statistical Parsing) which is another system designed for syntactic annotation of free text. [5] 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 specifying 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. [6] 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.

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 [6], 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.

```
Algorithm 2 - Grading Against Rule Set
 Inputs:
   Sentence or sentences of a summary set to be
   graded for their grammar.
 Outputs:
   Graded rating of how accurate the grammar is
   based on comparison with the systems existing
   rule set.
 ------------------------------------------------
 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) gradeSummary():
         for sequence in listOfTagSequences:
            if sequence is in in file:
                ■ Add +1 to total summary score
            end if
         end for
         ■ Divide total score by total # of sentences
         for a final grade
```

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 included 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 [4], 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

In the future, we hope to further decrease the false negative rate by implementing a system to weigh grammaticality on a scale, versus scoring it completely correct or incorrect. We think that grading on a scale, meaning a sentence can be partially correct, would decrease the average number of false negatives. In addition, we think it will be important to experiment with additional tagging systems, mentioned in Section 3.1. Currently, we are using a very simple tagset. Our hypothesis was that the more simple the tag set, the more general sequences we would get, meaning more overlap in rule sequences. After reporting only about a 5% overlap, we think that experimenting with more robust tagsets could have an significant impact on our results.

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] R. Vadlapudi and R. Katragadda, "Quantitative evaluation of grammaticality of summaries," *http://www.academia.edu*, 2010.

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

[5] 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.

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

Appendices

A Literature

Listed are the pieces of literature used in the created of the tag sequences file. Literature denoted with a "G" was obtained from Project Gutenberg. Anything denoted with a "P" indicates a book from one of the author's personal collection. The literature is ordered in the way it was added to the tag sequences file.

1. Carroll, Lewis. (2008). *Alice in Wonderland.* Urbana, Illinois: Project Gutenberg. Retrieved April 1, 2017, from http://www.gutenberg.org/ebooks/11. G.

2. Bronte, Charlotte. (1998). *Jane Eyre.* Urbana, Illinois: Project Gutenberg. Retrieved April 26, 2017, from http://www.gutenberg.org/cache/epub/1260/pg1260.txt. G.

3. Twain, Mark. (2004). *The Adventures of Tom Sawyer.* Urbana, Illinois: Project Gutenberg. Retrieved April 26, 2017, from http://www.gutenberg.org/files/74/74-0.txt. G.

4. Melville, Herman. (2001). *Moby Dick.* Urbana, Illinois: Project Gutenberg. Retrieved April 1, 2017, from http://www.gutenberg.org/files/2701/2701-0.txt. G.

5. Wilde, Oscar. (1994). *The Picture of Dorian Gray.* Urbana, Illinois: Project Gutenberg. Retrieved April 1, 2017, from http://www.gutenberg.org/cache/epub/174/pg174.txt. G.

6. Dickens, Charles. (1994). *A Tale of Two Cities.* Urbana, Illinois: Project Gutenberg. Retrieved April 26, 2017, from http://www.gutenberg.org/cache/epub/174/pg174.txt. G.

7. Gilman, Charlotte Perkins. (1999). *The Yellow Wallpaper.* Urbana, Illinois: Project Gutenberg. Retrieved April 26, 2017, from http://www.gutenberg.org/cache/epub/1952/pg1952.txt. G.

8. Austen, Jane. (1994). *Emma.* Urbana, Illinois: Project Gutenberg. Retrieved April 26, 2017, from http://www.gutenberg.org/files/158/158-0.txt. G.

9. Austen, Jane. (1998). *Pride and Prejudice.* Urbana, Illinois: Project Gutenberg. Retrieved April 1, 2017, from http://www.gutenberg.org/files/1342/1342-0.txt. G.

10. Austen, Jane. (1994). *Sense and Sensibility.* Urbana, Illinois: Project Gutenberg. Retrieved April 26, 2017, from http://www.gutenberg.org/cache/epub/161/pg161.txt. G.

11. Barrie, J. M. (2008). *Peter Pan.* Urbana, Illinois: Project Gutenberg. Retrieved April 28, 2017, from http://www.gutenberg.org/files/16/16-0.txt. G.

12. Wells, H. G. (2004). *The Time Machine.* Urbana, Illinois: Project Gutenberg. Retrieved April 28, 2017, from http://www.gutenberg.org/cache/epub/35/pg35.txt. G.

13. Stevenson, Robert Louis. (2006). *Treasure Island.* Urbana, Illinois: Project Gutenberg. Retrieved April 28, 2017, from http://www.gutenberg.org/cache/epub/120/pg120.txt. G.

14. James, Henry. (1995). *The Turn of the Screw.* Urbana, Illinois: Project Gutenberg. Retrieved April 28, 2017, from http://www.gutenberg.org/files/209/209-0.txt. G.

15. Rowling, J. K. (1997). *Harry Potter and the Sorcerer's Stone.* New York, NY: Scholastic Press. P.

16. Rowling, J. K. (1999). *Harry Potter and the Chamber of Secrets.* New York, NY: Scholastic Press. P.

17. Rowling, J. K. (1999). *Harry Potter and the Prisoner of Azkaban.* New York, NY: Scholastic Press. P.

18. Rowling, J. K. (2000). *Harry Potter and the Goblet of Fire.* New York, NY: Scholastic Press. P.

19. Rowling, J. K. (2003). *Harry Potter and the Order of the Phoenix.* New York, NY: Scholastic Press. P.

20. Rowling, J. K. (2005). *Harry Potter and the Half-Blood Prince.* New York, NY: Scholastic Press. P.

21. Rowling, J. K. (2007). *Harry Potter and the Deathly Hallows.* New York, NY: Scholastic Press. P.

22. Gaiman, Neil. (2010). *Stardust.* New York, NY: HarperTeen. P

23. Bradbury, Ray. (2008). *Fahrenheit 451.* Moline, IL: Moline Public Library. P.

24. Bradbury, Ray. (2014). *Something Wicked This Way Comes.* New York, NY: Harper. P.

25. Card, Orson Scott. (2003). *Speaker for the Dead.* Princeton, NJ: Princeton. P.

26. Collins, Suzanne. (2008). *The Hunger Games.* New York, NY: Scholastic Press. P.

27. Collins, Suzanne. (2009). *Catching Fire.* New York, NY: Scholastic Press. P.

28. Collins, Suzanne. (2010). *Mockingjay.* New York, NY: Scholastic Press. P.

29. Fitzgerald, F. Scott. (1995). *The Great Gatsby.* Ware: Wordsworth. P.

30. Gaimain, Neil. (2014). *Neverwhere.* New York, NY: Harper. P.

31. Card, Orson Scott. (2009). *Ender's Game.* New York, NY: Marvel Publishers. P.

32. Gaiman, Neil. (2003). *American Gods.* New York, NY: Perennial. P.

33. Card, Orson Scott. (2002). *Ender's Shadow.* New York, NY: Starscape. P.

34. Card, Orson Scott. (2002). *Children of the Mind.* New York, NY: Starscape. P.

35. Herbert, Frank. (2005). *Dune.* New York, NY: Ace Books. P.

36. Lowry, Lois. (1993). *The Giver.* Boston, MA: Houghton Mifflin. P.

37. McEwan, Ian. (2002). *Atonement: A Novel.* New York, NY: Nan A. Talese/Doubleday. P.

38. Larson, Erik. (2003). *The Devil in the White City: Murder, Magic, and Madness at the Fair that Changed America.* Princeton, NJ: Princeton. P.

39. King, Stephen. (1977). *The Shining.* New York, NY: Anchor. P.

40. King, Stephen. (2006). *Cell.* London: Hodder & Stoughton. P.

41. King, Stephen. (2004). *Carrie.* New York, NY: Doubleday. P.

42. King, Stephen. (2001). *Bag of Bones.* New York, NY: Scribner. P.

43. Hemingway, Ernest. (2003). *The Sun Also Rises.* New York, NY: Scribner. P.

44. Hemingway, Ernest. (1995). *A Farewell to Arms.* New York, NY: Scribner. P.

45. Huxley, Aldous. (2007). *Brave New World.* London: Grafton Books. P.

46. Irving, John. (2002). *A Prayer for Owen Meany: A Novel.* New York, NY: Modern Library. P.

47. Jordan, Robert. (2000). *The Eye of the World.* New York, NY: Tor Books, LLC. P.

48. Mitchell, Margaret. (2007). *Gone with the Wind.* New York, NY: Scribner. P.

49. Kingsolver, Barbara. (2007). *The Poisonwood Bible: A Novel.* New York, NY: Harper Perennial. P.