Consistency scores in text data *

Ke-Li Chiu *University of Toronto*
Rohan Alexander *University of Toronto*

In this paper we introduce a process to clean the text extracted from PDFs using n-gram models that is widely used in statistical natural language processing. Our approach compares originally extracted text with the text generated from, or expected by, these methods using earlier text as stimulus. To guide this process, we introduce the notion of a consistency score, which refers to the proportion of text that is unchanged by the model. This is used to monitor changes during the cleaning process, and to compare the messiness of different texts. We illustrate our process on text from the book Jane Eyre and introduce both a Shiny application and an R package to make our process easier for others to adopt.

Keywords: text-as-data; natural language processing; quantitative analysis.

Introduction

When we think of quantitative analysis, we may like to think that our job is to 'let the data speak'. But this is rarely the case in practise. Datasets can have errors, be biased, incomplete, or messy. In any case, it is the underlying statistical process of which the dataset is an artifact that is typically of interest. In order to use statistical models to understand that process, we typically need to clean and prepare the dataset in some way. This is especially the case when we work with text data. But this cleaning and preparation requires us to make many decisions. To what extent should we correct obvious errors? What about slightly-less-obvious errors? Although cleaning and preparation is a necessary step, we may be concerned about the extent to which have we introduced new errors, and the possibility that we have made decisions that have affected, or even driven, our results.

In this paper we introduce the concept of consistency in a text corpus. Consistency refers to the proportion of words that are able to be forecast by a statistical model, based on preceding words and surrounding context. Further, we define internal consistency as when the model is trained on the corpus itself, and external consistency as when the model is trained on a more general corpus. Together, these concepts provide a guide to the cleanliness and consistency of a text dataset. This can be important when deciding whether a dataset is fit for purpose; when carrying out data cleaning and preparation tasks; and as a comparison between datasets.

To provide an example, consider the sentence, 'the cat in the…'. A child who has read this book could tell you that the next word should be 'hat'. Hence if the sentence was actually 'the cat in the bat', then that child would know something was likely wrong.

*Thank you to **X, Y and Z** for helpful comments. A Shiny application for interactive and small-scale examples is available: https://kelichiu.shinyapps.io/Arianna/. An R package for larger scale applications is available: **HERE**. Our code and datasets are available: https://github.com/RohanAlexander/consistency_scores_in_text_datasets. Comments on the 31 August 2020 version of this paper are welcome at: rohan.alexander@utoronto.ca.

The consistency score would likely be lower than if the sentence were 'the cat in the hat'. After we correct this error, the consistency score would likely increase. By examining how the consistency scores evolve in response to changes made to the text during the data preparation and cleaning stages we can better understand the effect of the changes. Including consistency scores and their evolution when text corpuses are shared allows researchers to be more transparent about their corpus. And finally, the use of consistency scores allows for an increased level of automation in the cleaning process.

We employ n-gram models to calculate consistency scores and generate word candidates for text corrections. The n-gram approach involves identifying two, three, or more, words that are commonly found together. **(Keli adds descriptions on how n-gram can be used to calculate consistency scores and generate word candidates)**

The remainder of our paper is structured as follows: **(Keli add structure)**. Additionally, we construct a Shiny app available at: https://kelichiu.shinyapps.io/aRianna/. That app computes internal and external consistency scores for corpus excerpts, and have developed an R Package, `aRianna`, that allows our approach to be used on larger datasets, which is available at: https://github.com/RohanAlexander/aRianna.

Background

Language Model

Given the nature of human languages, some combinations of words tend to occur more frequently than others. Think of 'good', which is more often followed by 'morning', than 'duck'. As such, we could consider English text production as a conditional probability, $\Pr(w_k|w_1^{k-1})$, where k is the number of words in a sentence, w_k is the predicted word, and w_1^{k-1} is the history of the word occurring in a sequence (Brown et al., 1992). In this way, the generation of some prediction, w_k, is based on the history, w_1^{k-1}. This is the underlying principle of all language models. Essentially, a language model is a probability distribution over sequences of words. The goal of statistical language modeling is to estimate probability distributions over different linguistic units — words, sentences, and even documents (Bengio et al., 2003). However, this is difficult as language is categorical. If we consider each word in a vocabulary as a category, then the dimensionality of language becomes large (Rosenfeld, 2000). The reason that there is such a variety of statistical language models is that there are various ways of dealing with this fundamental problem.

N-gram Models

The foundation of an n-gram language model is the conditional probability set-up introduced above. An n-gram model is a probabilistic language model that predicts the next word in a sequence of words (Bengio et al. (2003)). The n in n-gram refers to the number of words in that sequence. Consider the following excerpt from *Jane Eyre*: 'We had been wandering'. 'We' is a uni-gram, 'We had' is a bi-gram, 'We had been' is a tri-gram, and 'We had been wandering' is a 4-gram. Notice that the two tri-grams in this excerpt, 'We had been', and 'had been wandering', overlap. The use of n-gram models enables

us to assign probabilities to both the next sequence of words and just the next word. For instance, consider the two sentences: 'We had been wandering', and 'We had been wangling'. The former is likely to be more frequently encountered in a training corpus. Thus, an n-gram would assign a higher probability to the next word being 'wandering' than 'wangling', given the sequence 'We had been'.

To predict the next word, we have to take the sequence of preceding words into account, which requires knowing the probability of the sequence of words. The probability of a sequence appearing in a corpus follows the chain rule:

$$\Pr(\text{We,had,been,wandering}) = \Pr(\text{We}) \times \Pr(\text{had} \mid \text{We}) \times \Pr(\text{been} \mid \text{We,had}) \times \Pr(\text{wandering} \mid \text{We,had,bee}$$

However, the likelihood that more and more words will occur next to each other in an identical sequence becomes smaller and smaller, making prediction difficult. Alternatively, we can approximate the probability of a word depending on only the previous word. This is known as the 'Markov assumption' and it allows us to approximate the probability using only the last n words (Brown et al., 1992):

$$\Pr(\text{We,had,been,wandering}) \approx \Pr(\text{We}) \times \Pr(\text{had} \mid \text{We}) \times \Pr(\text{been} \mid \text{had}) \times \Pr(\text{wandering} \mid \text{been}).$$

As a bi-gram model only considers the immediately preceding word, under the Markov assumption, an n-gram model can be reduced to a bi-gram model with n being any number:

$$\Pr(w_n|w_1^{n-1}) \approx \Pr(w_n|w_{n-1})$$

Language models underpinned by n-grams are widely applied in text prediction, spelling-correction, and machine translation (Brown et al., 1992). In our application, we use a tri-gram based model to detect both real-word and non-word errors and correct them with the candidate word that has the highest probability.

Package Dependencies

There are a variety of ways to implement an n-gram model within R R Core Team (2019) including using R packages such as `Quanteda` (Benoit et al., 2018), `tidyText` (Silge and Robinson, 2016) and `tm` (Feinerer and Hornik, 2019). In our package, we used the `Quanteda` R package because it has a comprehensive set of functions for conducting text analysis. We also employed `dplyr` (**Keli adds citation**) for data frame manipulation and `tibble` (**Keli adds citation**) for transforming data frames to the tibble format:

Stylized example

As a stylized example, let's consider the following actual paragraph from Jane Eyre, by Charlotte Bronte:

> There was no possibility of taking a walk that day. We had been wandering, indeed, in the leafless shrubbery an hour in the morning; but since dinner

> (Mrs. Reed, when there was no company, dined early) the cold winter wind had brought with it clouds so sombre, and a rain so penetrating, that further out-door exercise was now out of the question.

Let's pretend that this text had been created from optical character recognition and that it had the following errors: some 'h' were replaced with 'b'; and some 'd' have been replaced with 'al':

> There was no possibility of taking a walk that day. We had been wandering, indeed, in tbe leafless shrubbery an hour in tbe morning; but since dinner (Mrs. Reeal, when tbere was no company, dined early) the cold winter wind had brought with it clouds so sombre, and a rain so penetrating, tbat further out-door exercise was now out of the question.

Assume a model that is trained to perfectly forecast the next word in Jane Eyre. For this fragment there are 62 words, comprising 5 errors and 57 correct words. So the internal consistency score of this fragment would be: $57/62 = 0.919$. When we recognise and correct the errors, this consistency score would increase to 1.

Similarly, assume a model that is trained on an external data source. This means that it will recognise the the cases where some 'h' were replaced with 'b', but not recognise that 'Reed' has become 'Reeal'. Hence, the external consistency score would be $58/62 = 0.935$.

Application

(Keli adds application in OCR tasks)

[Internal consistency: ground truth of the text and the text that are aligned with ground truth]

Discussion

(Keli organizes these text to illustrate limitations and future works)

Internal validity vs external- one is their own words the other is a general set of words.

In the same way that precision and recall provide important measures...

However, n-gram models do not take the linguistic structure of language into account. For instance, Rosenfeld (2000, p. 1) discusses language in this context, saying that '...it may as well be a sequence of arbitrary symbols, with no deep structure, intention or thought behind'. The prediction of the next word is based on only a few preceding words and broader context is not taken into account. Hence next-word prediction using n-gram based language models can be limited.

Here think of a two-gram involving the word 'good' 'good morning'. At scale these can identify missing or unusual words, and work quickly, but they lack nuance. For instance, an equally reasonable two-gram involving the word 'good' is 'good work'. For that reason, we consider pre-trained word embedding models. These include Word2Vec

and GloVe, which place each word in a multi-dimensional space such that distance between words can illustrate their relationship. We also consider pre-trained generative models such as GPT-2 and GPT-3 and BERT. GPT-2, GPT-3 and BERT are pre-trained generative unsupervised language models. GPT-2 and GPT-3 are the two generations of the same model that differ in the number of parameters. GPT-2 has 1.5 billion parameters and GPT-3 has more than 175 billion parameters. While GPT-2 is more limited, it can be run on smaller machines, whereas GPT-3 cannot and is accessed via an API. BERT has 340 millions parameters which is smaller than both GPT-2 and GPT-3. BERT also differs from GPT models because its encoding process is bidirectional instead of unidirectional.

References

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Benoit, K., Watanabe, K., Wang, H., Nulty, P., Obeng, A., Müller, S., and Matsuo, A. (2018). quanteda: An r package for the quantitative analysis of textual data. *Journal of Open Source Software*, 3(30):774.

Brown, P. F., Della Pietra, V. J., Desouza, P. V., Lai, J. C., and Mercer, R. L. (1992). Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–480.

Feinerer, I. and Hornik, K. (2019). *tm: Text Mining Package*. R package version 0.7-7.

R Core Team (2019). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rosenfeld, R. (2000). Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88(8):1270–1278.

Silge, J. and Robinson, D. (2016). tidytext: Text mining and analysis using tidy data principles in r. *JOSS*, 1(3).