Corpus-based Error Detector for Computer Scientists

John Blake
University of Aizu
Aizu-wakamatsu, Japan.
jblake@u-aizu.ac.jp

Abstract

This study describes the design and development of a corpus-based error detector for short research articles produced by computer science majors. This genre-specific error detector provides automated pedagogic feedback on surface-level errors using rule-based pattern matching. In the corpus phase, a learner corpus of all theses (n = 629) submitted for three academic years was compiled. A held-out corpus of 50 theses was created for evaluation purposes. The remaining theses were added to the working corpus. Errors in the working corpus were identified manually and automatically. The first 50 theses were annotated using the UAM Corpus Tool. Errors were classified into one of five categories (i.e. accuracy, brevity, clarity, objectivity and formality). By the fiftieth thesis, saturation had been reached, that is the number of new errors discovered had dropped considerably. Annotated errors were extracted into an error bank (xml file). Each error was assigned values for severity, detectability and frequency. The weighted priority of each error was calculated from these values. For the remaining theses only new errors were recorded and were added directly into the error bank. In the software phase, regular expressions were created. Easy-to-understand actionable advice was written that could be displayed on matching the error.

Keywords: learner corpora, error detection, computer science

1. Introduction

1.1 Background

All students at the University of Aizu are required to submit a thesis in English to fulfil graduation requirements. The university specializes in computer science and so the thesis for undergraduates takes the form of a computer science research article. The format and style of the article replicates IEEE journals, but the requirements of originality, significance and substance are less rigorous than peer-reviewed journals. The thesis serves as a vehicle for students to learn rather than to contribute to the research literature.

Undergraduate students face two language-related problems when writing their thesis: (1) lack of proficiency in written English, and (2) lack of familiarity with the genre of formal scientific articles. Students are offered a course entitled "Thesis writing and presentation" in the final semester of their senior year. The primary aim of this course is to enable students to complete their theses. Students on the course regularly submit small sections of their thesis to their teacher for comments. Teachers often note that many of the mistakes that they provide feedback on are predictable surface-level mistakes.

1.2 Purpose

This study describes the design and development of a corpus-based error detector for short research articles produced by computer science majors. The error detector provides automated pedagogic feedback using rule-based pattern matching on surface-level errors that were discovered in a corpus.

1.3 Overview

This paper focusses on the use of corpus linguistics in the development of this error detector. The following section introduces how rule-based pattern-matching can be used to detect errors. Section three describes the design of the corpus while section four focuses on corpus annotation. The fifth section describes the corpus analysis stage and the protocol for prioritizing errors to be incorporated into the detection tool. Section six gives an overview of the software development phase. The paper concludes with a brief discussion of the usability, accuracy and efficacy of the error detector.

2. Error Detection

2.1 Rule-based pattern matching

There are two main ways of automated error detection namely probabilistic parsing and rule-based pattern matching searches. Although probabilistic parsing algorithms are very effective, rule-based pattern matching was selected due to its simplicity and ease of deployment.

Patterns permeate language (Kilgarriff, 2005) with certain words tending to co-occur with other particular words, or in the frequently quoted words of Firth (1957), "You shall know a word by the company it keeps" (p.11). This is also the case for the interlanguage of learners of English. Novice users of a language tend to make predictable patterns of mistakes as evidenced by the plethora of published pedagogic books that aim to help Japanese learners avoid making such mistakes (Barker, 2003; Barker, 2008; Ishikawa, 2008; Webb, 2006).

The grammar checker within Microsoft Word and many other generic grammar checkers identify patterns using regular expressions. Most checking software that harnesses rule-based matching automatically identifies mistakes and suggests a replacement or automatically replaces the matched item. This autocorrect-type function may increase the quality of the product, i.e. the writing; but may not help the writer understand the underlying reasons for the suggested changes. The result may be that many novice writers accept all suggestions without trying to assess whether the results are true or false positives.

Many of these systematic irregularities in the interlanguage of language learners are easily discovered by regular expressions. Regular expression searches can be made for items that conform to a grammatically inaccurate rule and any items discovered may be identified as errors.

To the best of my knowledge, the first online error detector for specific purposes was created by Morrall (2000) to provide student writers at the Hong Kong Polytechnic University with automated advice on how to write academic essays. The mistakes that were coded for were based on typical errors that were submitted in assessments. This detector provides two types of feedback: *errors* and *warnings*. The term *error* is used when the developer is positive that the matched expression is incorrect, while *warning* is used when the matched expression may be incorrect.

2.2 Regular expressions

Regular expressions are similar to wildcard searches, but are more refined and can find complex combinations of characters, e.g. letters and words (see Friedl, 2006; Watt, 2005 for comprehensive introductions). Regular expressions are commonly used by corpus linguists, especially when cleaning web-crawled corpora. Particular sequences of characters, such as urls or page numbers, can easily be removed or replaced.

In this tool, regular expressions are used to match expressions to help users learn about potential errors in their draft. The following example shows how this is achieved. Consider sentence (1) which contains a typical grammatical error. This sentence could be revised by replacing *to* with *and* as shown in sentence (2). To discover this automatically, the regular expression in (3) can be used in a Javascript function. This expression searches for the words *between* and *to* when either one or two wo. matches the words enclosed in the box in sentence (4). On matching a script is automatically executed than generates a feedback message, such as one shown in (5).

(1) There were between 20 to 30.
(2) There were between 20 and 30.
(3) /\bbetween\W+(?:\w+\W+){1,2}?to\b/gi;
(4) There were **between 20 to** 30.
(5) Use "between X and Y."

For specific genres with high generic integrity (Bhatia, 1996), it is possible to target user errors more easily and more accurately. Unsuitable phraseologies can be ruled out. For example, the expression "There happened" could begin these sentences:

(6) There happened to be a solution.
(7) There happened a problem in the software.*

Sentence (6) is grammatically accurate, but is highly unlikely to be found in graduation theses of computer science majors, while sentence (7) in which the intransitive verb *happen* is used transitively commonly occurs in learner English. The asterisk denotes that this form deviates from the grammar expected in Anglophone countries, i.e. the inner circle of Kachru's three-circles-of-English model of world Englishes (Kachu, 1992).

Creating, checking and debugging arrays (sets) of regular expressions is time-consuming, and so to maximize cost performance, prioritizing which errors to target is a priority. Given the assumption that future cohorts of writers are likely to make similar mistakes, an analysis of errors within a corpus can be used to predict which errors are likely to occur in the writings of future cohorts of scientific writers.

3. Corpus development

The corpus development phase generates the data needed for the software development. Although it is possible to adopt an armchair linguist approach and imagine which errors learners make, a more scientific, replicable approach is to base decisions on concrete evidence. In this case, a corpus of the target genre: graduation theses.

3.1 Corpus specification

A corpus of all theses submitted over three-year period was selected. With approximately two hundred students graduating each year, this would provide a corpus of approximately 600 theses. Each thesis is expected to be between four and six pages assuming that the theses comply with university guidelines.

3.2 Corpus collection

In the corpus collection phase, a learner corpus of all theses (n = 629) submitted for three academic years (AY 2014 to AY 2016) was compiled. A held-out corpus of 50 theses was created for evaluation purposes. The remaining theses (n = 579) were added to the working corpus. The next phase is to annotate the corpus, pinpointing and classifying the errors.

4. Corpus annotation

4.1 Manual annotation

Errors in the working corpus were identified manually by skimming and scanning the texts. The corpus is far larger than can be annotated intensively by one researcher within the timeframe set to complete the annotation stage (four months).

4.2 Error taxonomy

Errors were classified into one of five categories, mirroring the content of the University of Aizu in-house thesis writing course. Table 1 is used in the thesis writing course to show novice writers the key criteria to evaluate the language used in their graduation theses. These same criteria are used for proofreading purposes, enabling learners to systematically review their writing focussing on one criterion at a time.

Type	Description
Accuracy	Factual and language errors
Brevity	Using too many words
Clarity	Using vague or ambiguous terms
Objectivity	Using terms that appear subjective
Formality	Using abbreviations, contractions, and informal terms

Table 1 Key criteria for research writing

The five categories are designed as pedagogic categories that can help learners edit their language.

These five categories were taken from an error classification scheme created for an earlier study of a corpus of 200 draft research articles submitted for internal review by graduate students at the Japan Advanced Institute of Science and Technology (Blake, 2016). In that

study, template analysis (King, 2004) of the pedagogic literature on scientific writing was conducted which uncovered three major criteria: accuracy, brevity and clarity; and two minor criteria: objectivity and formality.

4.3 Five error types

Accuracy is important, but actual accuracy is paramount. Thesis supervisors who grade theses may overlook language errors, but are seldom tolerant of subject-specific factual errors. However, language errors can be intrusive. at times affecting subject-specific meanings.

Brevity is used to refer to the removal of redundant and repetitive expressions. There is a trade-off between brevity and clarity. Audience awareness is central to achieving the optimum balance so that the intended message can be conveyed in the minimum number of words. Readers rely on co-text, context and world knowledge to decode messages.

Clarity of expression focuses on precision and removal of ambiguous and vague expressions. Ambiguity can be further divided into lexical, structural or referential, and any of these may lead to the unintended presence of garden-path sentences.

Objectivity is used to refer to a reduction in appeared subjectivity. Research articles in computer science focus on the process and products of research, and not on people and feelings.

Formality is a nebulous category that impinges on multiple other categories. Formality can be considered an aspect of style or register. Gilquin and Paquot (2008) note that the academic writing of learners of English tends to be rather informal. The following example shows how formality and objectivity are entwined.

(1) They developed an online tool.
(2) An online tool was developed.
(3) an online tool's development
(4) Development of an online tool

In sentence (2) the removal of the person deixis (doer of the action) and use of passive voice increases the formality and (arguably) objectivity compared to sentence (1). The nominalization or use of grammatical metaphor (Halliday, 1985) reduces the time deixis, creating a less context-dependent abstraction in sentence (3). However, the use of an apostrophe appears rather marked since *of* tends to be used with inanimate nouns. The unmarked formal version is shown in sentence (4).

4.4 Annotation tool

The first batch of 50 theses was annotated using the UAM Corpus Tool (O'Donnell, 2008). This tool was selected based on the ease of creating a tailor-made annotation scheme. The graphical scheme editor was used to create a tailor-made scheme. The annotations were completed at a finer level of granularity using the error classification scheme shown in Figure 1.

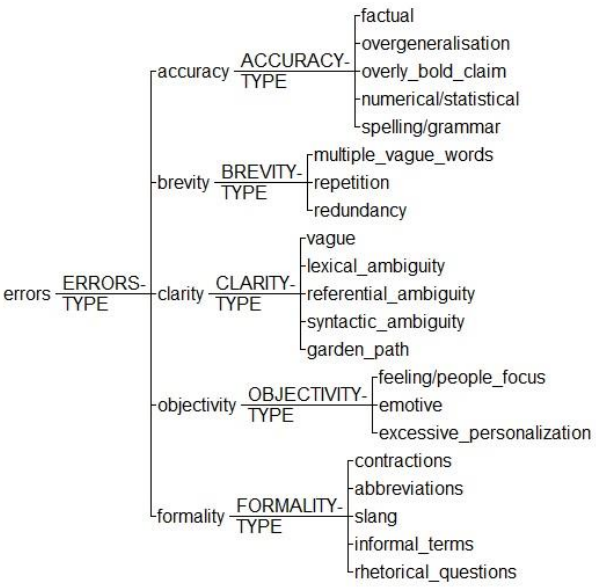

Figure 1 Error classification scheme

4.5 Saturation

By the fiftieth thesis, saturation had been reached, that is the number of new errors discovered had dropped considerably. Error frequency appears to follow Zipf's Law (Cancho and Solé, 2001). The cost-benefit for further annotations was not judged as being viable, and so no more detailed annotations were completed. However, exploratory regular expression searches were conducted to identify errors in the remaining theses in the working corpus and assess the frequency of the types of errors identified.

5. Corpus analysis

Annotated errors were extracted into an error bank. Using a failure mode effects analysis framework (Stamatis, 2003), each error was assigned values for severity, detectability and frequency. Intrusive errors affecting meaning were classified as severe. Detectability was estimated based on the perceived difficulty of creating a regular expression to match the error. Frequency was counted or estimated by investigating the occurance of the error in the whole working corpus. The weighted priority of each error was calculated from these values with severe, detectable, frequent errors receiving the hightest weighting.

6. Software development

This section outlines some of the steps in the software development phase that are of more interest to corpus linguists.

6.1 Creation of regular expressions

Regular expressions were created to detect the errors starting with those assigned the highest weighted priority.

6.2 Feedback messages

Easy-to-understand actionable advice was written that could be displayed on matching the error. These messages were written in English, but given that most users will be Japanese speakers, the option to select the feedback language may be added in a later version.

6.3 Interface development

A user-friendly interface was created and tested. This version of the interface uses toggle buttons, and when the cursor hovers over a button, an explanation of the purpose of the button appears as shown in Figure 2. The border of the toggle buttons are coloured using the same scheme as the inline feedback. Users can select to display one or more types of error simultaneously. Users with numerous errors are advised to focus on each error type, in turn.

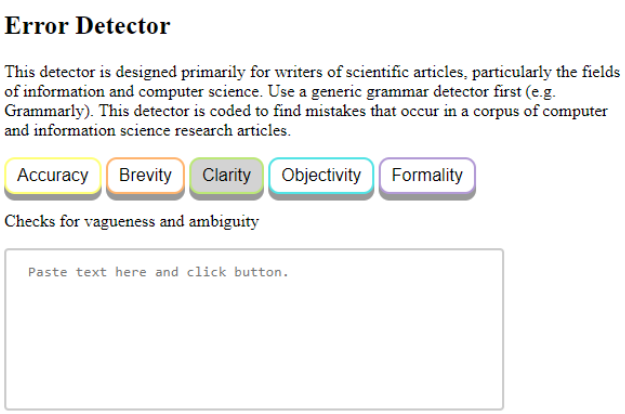

Figure 2 Interface for the Error Detector

7. Discussion

7.1 Summary

The Error Detector reduces the need for teachers to provide feedback on commonly-occurring surface-level errors. The error detector finds numerous errors in each writing activity conducted in the thesis writing course. Students are able to not only identify their own errors using the tool, but receive actionable advice on how to resolve the errors, and where appropriate, brief explanations are given to help learners avoid making the same type of error again.

Although the Error Detector is still under development, it is fully functional and able to discover numerous errors in each draft thesis submitted to date. This helps both the students and their teachers. Students are able to get instant feedback on potential errors in their thesis, and teachers no longer feel obliged to check for the types of mistakes that can be automatically identified.

7.2 Preliminary evaluation

7.2.1 Usability

Usability studies were conducted with small cohorts of undergraduate students. Numerous incremental changes were made of the development of the interface based on comments received.

7.2.2 Accuracy

The accuracy of each regular expression was tested to minimize the number of false positive results and maximize the number of true positive results. A formal evaluation of the accuracy of the Error Detector will be conducted when all the regular expressions have been added to the code.

7.2.3 Efficacy

The Error Detector is able to identify the most frequent genre-specific errors. Figure 3 shows the output when a student checked a draft of an introductory paragraph. The vast majority of students in the University of Aizu who use the Error Detector find many more mistakes in the accuracy category.

7.3 Further work

There are four features that are currently under development. First, more regular expressions and feedback messages are being created in order of weighted priority for the errors extracted from the corpus. Second, a bank of short screencast PowerPoint explanation videos recorded in Japanese have been created. A hyperlink will be added to the feedback message to provide users with the choice to receive a more detailed explanation. Third, the disambiguation of some expressions is either impossible or too time-consuming without having access to the part of speech of the words. A part-of-speech (POS) tagger, the nlp compromise Javascript library, (Kelly, 2016) is currently under trial. This tagging enables finer tuning, or disambiguation, of error detection since combinations of both words and POS tags can be matched. Fourth, an improved user interface based on the Grammarly interface that is designed to enhance the user experience is underdevelopment.

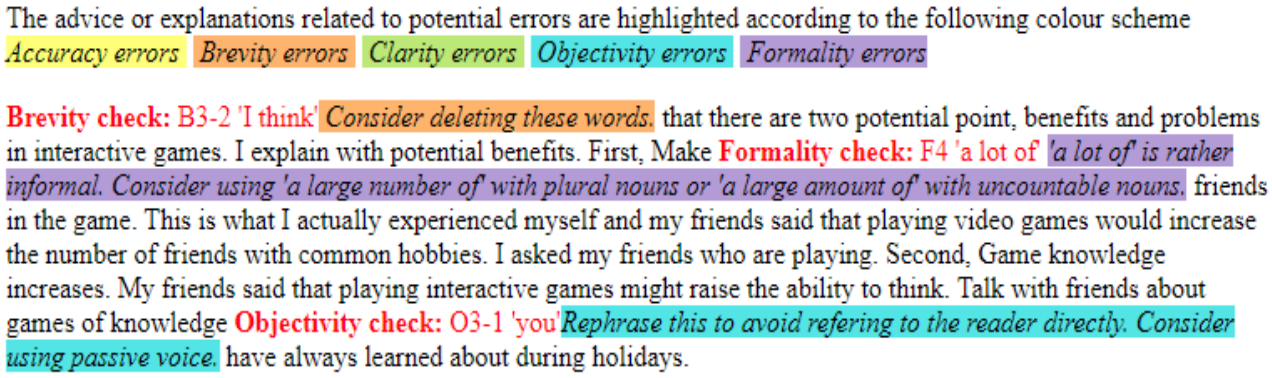

Figure 3 Screenshot of automated output

8. References

Barker, D. (2003). *Eigo to Nakanaori Dekiru Hon* [The Book for Becoming Friends Again with English]. Tokyo: Aruku.

Barker, D. (2008). *An A - Z of Common English Errors for Japanese Learners (Japanese version).* Tokyo: BTB Press.

Bhatia, V. K. (1996). Methodological issues in genre analysis. *Hermes, 16*, 39-60.

Blake, J. (2016) Pedagogic application of regular expressions: Corpus-based online writing tool. Paper presented at the 2nd BAAL Corpus linguistics SIG Event, 10 December 2016. Coventry University, England.

Ferrer i Cancho, R., & Solé, R. V. (2001). Two Regimes in the Frequency of Words and the Origins of Complex Lexicons: Zipf's Law Revisited∗. *Journal of Quantitative Linguistics, 8*(3), 165-173.

Firth, J.R. (1957). A synopsis of linguistic theory 1930-55. Reprinted in E.R. Palmer (Ed.), (1968), *Selected Papers of J.R. Firth 1952-59.* London: Longman.

Friedl, J. E. (2006). *Mastering Regular Expressions.* Sebastopol, CA: O'Reilly Media.

Gilquin, G., & Paquot, M. (2008). Too chatty: Learner academic writing and register variation. *English Text Construction, 1* (1), 41-61. Available online: http://dx.doi.org/10.1075/etc.1.1.05gil

Halliday, M. A. K. (1985). *An Introduction to Functional Grammar*. London: Arnold.

Heylighen, F. & Dewaele, J.-M. (1999). Formality of language: definition, measurement and behavioral determinants. *Internal Report, Center Leo Apostel, Free University of Brussels* .

Hunston, S. (2002). *Corpora in Applied Linguistics. Cambridge* : Cambridge University Press.

Ishikawa, S. (2008). *Eigo koupasu to eigo kyouiku: deeta toshiteno tekusuto. [Engish corpus and language education: Text as data]*. Tokyo: Taishukanshoten.

Kachru, B. (1992). *The Other Tongue: English across cultures*. University of Illinois Press.

Kelley, S. (2016). Language as an Interface. Presentation at "goto;" conference in Chicago (24-25 May 2016), Available online:: https://gotocon.com/dl/goto-chicago2016/slides/SpencerKelley_LanguageAsAnInterface.pdf.

Kilgarriff, A. (2005). Language is never ever ever random. *Corpus Linguistics and Linguistic Theory 1*, 263-276.

King, N. (2004). Using templates in the thematic analysis of text. In C. Cassell & G. Symon (Eds.), *Essential guide to qualitative methods in organizational research (pp. 256–270).* London: Sage. Available online: http://dx.doi.org/10.4135/9781446280119.n21

Leech, G. (2005). Adding Linguistic Annotation. In M. Wynne, (Ed), *Developing Linguistic Corpora: A Guide to Good Practice (pp.17–29).* Oxford: Oxbrow Books.

Morrall, A. (2000). Common error detector [online tool] Available online: http://www2.elc.polyu.edu.hk/cill/errordetector.htm.

McEnery, T. & Wilson, A. (2001). *Corpus Linguistics.* Edinburgh: Edinburgh University Press.

O'Donnell, M. (2008, April). The UAM CorpusTool: Software for corpus annotation and exploration. In *Proceedings of the XXVI Congreso de AESLA (pp. 3-5).* Almeria Spain.

Stamatis, D. H. (2003). *Failure Mode and Efect analysis: FMEA from Theory to Execution*. ASQ Quality Press.

Swales, J. M. (2004). *Research Genres: Explorations and Applications.* Cambridge: Cambridge University Press.

Watt, A. (2005). *Beginning Regular Expressions.*Indianapolis, IN: Wiley Publishing.

Webb, J. H. M. (2006). *151 Common Mistakes of Japanese Students of English*. Tokyo: The Japan Times.