Correct-Annotator: An Annotation Tool for Learner Corpora

Felix Hultin
Department of Linguistics
Stockholm University, Sweden
felixhultin@yahoo.se

Abstract

This paper presents CORRECT-ANNOTATOR, a brower-based, single-page annotation tool to annotate language errors for learner corpora in Swedish. Having grown out of the research project SweLL, CORRECT-ANNOTATOR attempts to significantly ease the workflow of an annotator, by allowing the annotator to edit and correct the text, from which the system induces potential language error annotations. This differs from previous annotation tools in that annotation is made on text transformations, rather than on a static text. With the expansion of learner corpora-related research in recent years, CORRECT-ANNOTATOR might prove useful tool for building learner corpora, which would be a unique addition to the CLARIN infrastructure.

1 Introduction

Software tools for corpus annotation have for a long time, in one form or the other, accompanied corpus building. Everything from annotating video and audio streams, such as ELAN (Sloetjes and Wittenburg, 2008), to word senses, such as the manual tagger user interface used in WordNet (Langone et al., 2004), have aided the linguist in creating structured corpora. Many are tied to specific corpus projects, while some, like BRAT (Stenetorp et al., 2012), have made it their mission to be as general as possible, in order to be applicable for a wide range of projects.

All of these software tools have been developed for the sole purpose of easing the workflow of annotators, making the result less error-prone and the annotation process faster. What these annotation tools have had in common, is that they present the corpus as a static read-only text, of which the annotator's job is to add extra layers of structured annotation. CORRECT-ANNOTATOR challenges this sort of workflow, by 1) letting the annotator change the text content, from which potential language errors are induced and 2) making text correction itself the main part of the annotation work flow. Hence, CORRECT-ANNOTATOR annotates text transformations, rather than static text elements.

The need for this software grew out of the Swedish Learner Language corpus (SweLL) project, which aims to create an infrastructure for research of Swedish as a second language (Volodina et al., 2016). A part of this project is to create an L2-corpus consisting of 600 texts with annotated language errors. The corpus is a type of parallel corpus, in that a corpus of language learner text is corrected, creating a second *normalized* text, with a set of annotated language errors, defining the relationship of the edits between the two. This annotation goes through two steps, the first being to minimally edit the text in order to get a grammatically correct text, and the second being to edit the grammatically corrected text to such an extent that it fulfills certain formal style. Considering the complexities of the second step, the program presented in this paper is exclusively concerned with the first step of annotation.

Second language learner corpora is a rather new phenomena in corpus linguistics, having started in the late 1980's with the commercial Longman Learner's Corpus and gaining ground in the research community in the 1990's and 2000's (Sinclair, 2004, p. 128). Although most corpora have been of second language learners of English, such as the Uppsala Student English Corpus (Axelsson, 2000) and notably the International Corpus of Learner English (Granger et al., 2002), other languages have also been dealt with, such as German (Siemen et al., 2006), Spanish (Lozano, 2009), French (Granger, 2003), Swedish (Hammarberg, 2013) and recently Portugese (del Río et al., 2016). Again, as far as the writer of this paper knows, none of these projects have used a text transformation annotation tool like CORRECT-ANNOTATOR in their corpus building. In the light of this, CORRECT-ANNOTATOR might prove applicable in many more language learner corpus building projects and would be a unique supplement to already existing annotation tools, for example WebAnno (de Castilho et al., 2014), in the CLARIN infrastructure (Hinrichs and Krauwer, 2014).

2 Method And Technologies

In this section specific technologies, implementation methods and the algorithm/heuristic to identify language errors in CORRECT-ANNOTATOR will be presented. By language error, I largely refer to a linguistic construction made by a second language learner, which deviates from the rules of the target language. These might be, for example, spelling errors, grammatical errors, word order errors or morpheme errors, however, since there yet hasn't been any comprehensive error type list set out for the SweLL project (see the end of section 2.3.1), this remains an open question.

2.1 Technologies

In order for the annotation tool to be as accessible as possible, it was decided early on that the tool would be a browser-based application, more precisely a single-page application. This means that the tool is completely independent of operating system or any additional software, but a modern web browser (for example Google Chrome, Mozilla Firefox or Internet Explorer). Seeing that the program will most likely be used by professional (computational) linguists, situated in working environments with limited flexibility when it comes to using certain operating system or install additional software, this seemed to be the most flexible solution.

Beside common browser-related technologies (HTML, CSS and JavaScript), the JavaScript framework Backbone.js (Ashkenas, 2010), bundled with jQuery (Resig, 2017), was used. Backbone.js helps developers to minimally structure their web-application in a model-view-presenter design paradigm, where HTML can be bound to so-called views, data stored in so-called models and events handled in a modular approach.

The application consists of three main graphical interface components:

Text Editor A text editor, where the annotator can edit and correct the language errors of the text.

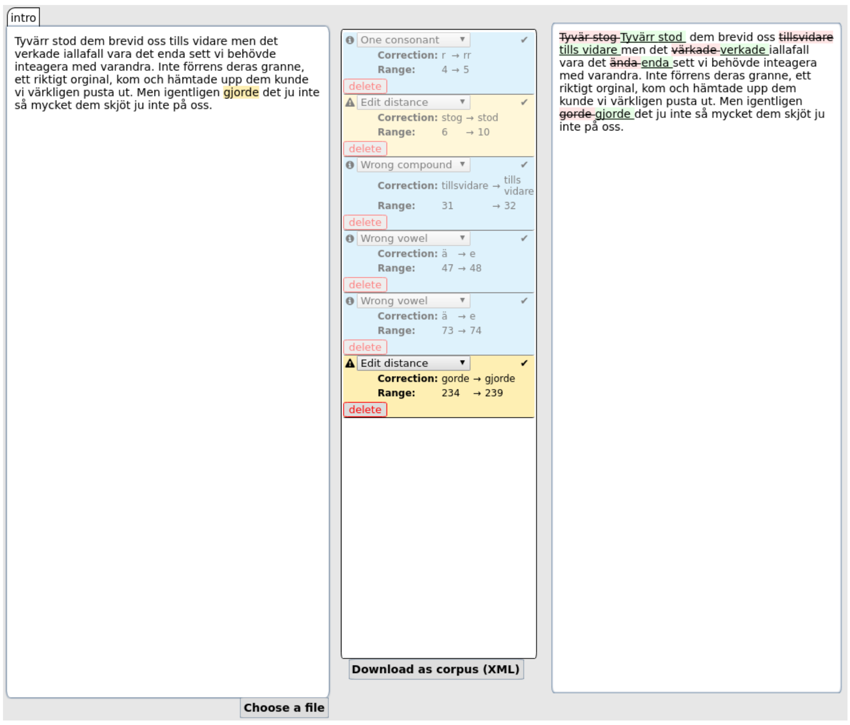
Language Error Table A list of induced language errors, as they will appear in the corpus data.

Diff viewer A diff view of the original text and the revised text, visualized in terms of deletions green and insertions red.

The usual workflow of an annotator would be to correct the language errors in the text, analyze the induced language errors and change their attributes if needed. To aid in this process, the language errors are colored according to the certainty of the correction. For example, if the vowel *e* in the wrongly spelled word *merker* was replaced with the other vowel *ä*, the registered language error would be colored blue, since the probability of the error resulting from phonological similarity is rather large. However, if a change generates the fallback language error, it would be colored red because no other language error could be induced. If the annotator edits the same word at the same language error position, a new induced language error would replace the old. However, if the annotator agrees with the language error, he/she can submit the error, by pressing the ✓-button, coloring the language error green, as in success, forcing the system not to remove it. The induced language errors can be assigned one of the four information statuses: error red, warning yellow, information blue, and success green. Certain attributes of the language errors can be edited, for example error type, however, static attributes, like the the position of the language error, cannot be edited.

2.2 Language Error Algorithm

Leaving the graphical user interface components aside, the core of CORRECT-ANNOTATOR is the algorithm, which induces the language errors. The algorithm is event-driven, in that it parses and induces language error from user interaction events:



Tyvärr stod dem brevid oss tills vidare men det verkade iallafall vara det enda sett vi behövde inteagera med varandra. Inte förrens deras granne, ett riktigt orginal, kom och hämtade upp dem kunde vi värkligen pusta ut. Men igentligen gjorde det ju inte så mycket dem skjöt ju inte på oss.

One consonant
Correction: r → rr
Range: 4 → 5

Edit distance
Correction: stog → stod
Range: 6 → 10

Wrong compound
Correction: tillsvidare → tills vidare
Range: 31 → 32

Wrong vowel
Correction: ä → e
Range: 47 → 48

Wrong vowel
Correction: ä → e
Range: 73 → 74

Edit distance
Correction: gorde → gjorde
Range: 234 → 239

~~Tyvär stog~~ Tyvärr stod dem brevid oss ~~tillsvidare~~ tills vidare men det ~~värkade~~ verkade iallafall vara det ~~ända~~ enda sett vi behövde inteagera med varandra. Inte förrens deras granne, ett riktigt orginal, kom och hämtade upp dem kunde vi värkligen pusta ut. Men igentligen ~~gorde~~ gjorde det ju inte så mycket dem skjöt ju inte på oss.

Figure 1: Screenshot of CORRECT-ANNOTATOR in action. After having edited at five different places in the text, six language errors have been identified.

As illustrated in figure 2, the program listens to user interaction events, such as keypress, cut or paste, which are triggered whenever the input of the text editor is changed. When triggered, the event is passed through a heuristic, where the event is tested against a set of language error types. Each language error type has defined conditions, which if the event, and sometimes previous events in the event history, match, returns a language error annotation. It contains information about the error type, the local correction made and the position of the error in the original text. Take our previous spelling error *merker* of the word *märker* as an example. As the user inserts the vowel *ä* in the place of *e*, the language error's condition will be satisfied, namely that if the event is a keypress of a vowel at a position where there was another vowel in its original string, the appropriate language error will be returned and the heuristic loop will break.

2.3 Discussion

The annotation tool presented in this paper shows a different approach to corpus annotation. Rather than the text being a static component in the user interface, to which the annotator can only add extra layers of annotations, the text in this tool is an editable entity, whose user operations helps in generating the annotated output. Although this tool is restricted to the specific task of annotating second language learner texts, it might inspire alternative ways of annotating natural language data.

Although no user studies have been made to prove it, one might expect that the annotation time would be much reduced. Because the program emulates the process of a person correcting a second language learner text in general, one can expect that the amount of user operations would drastically reduce. After all, the annotator only has to correct the text, verify the errors and, at most, change some attributes, while in a traditional annotation tool, like brat (Stenetorp et al., 2012), one would have to go through multiple user operations of double clicking a word element, enter error type, and multiple clicks to assign each non-static attribute. To further speed up the process, other external language correction programs, like the spelling correction program STAVA (Kann et al., 2001), could also be used by CORRECT-ANNOTATOR,

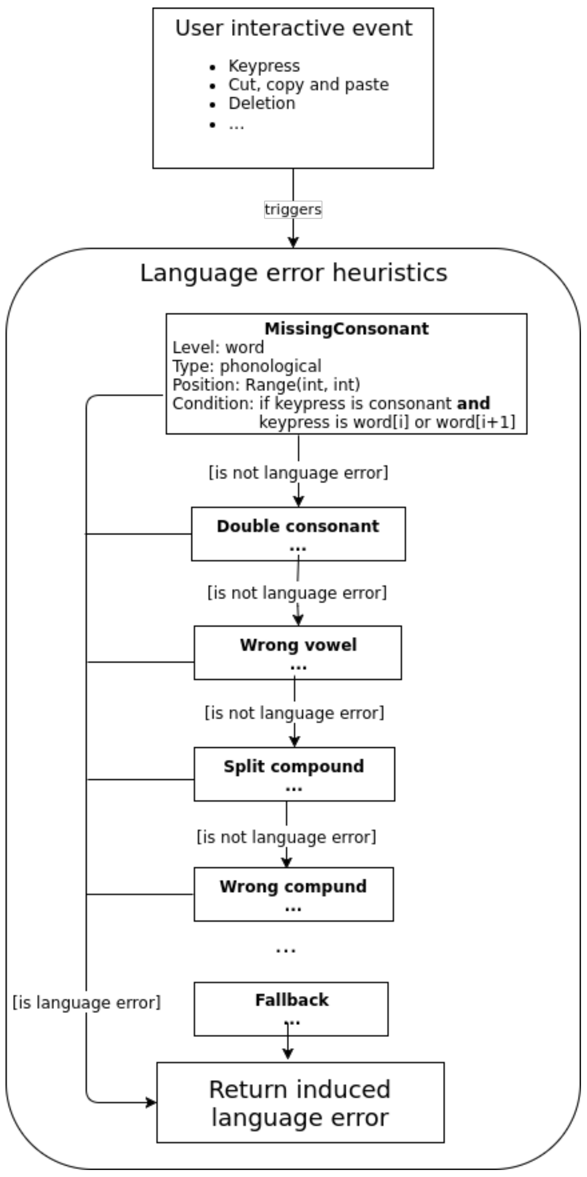


Figure 2: Algorithm and heuristic used to identify language errors in CORRECT-ANNOTATOR.

so that the annotator at many times does not even have to edit the text, but only delete or confirm the corrections made by the external program.

Seeing that the area of second language learner corpora is expanding and gaining traction within the research community, software like CORRECT-ANNOTATOR might prove critical in more efficiently building future learner corpora. Furthermore, with its current implementation and the software architecture laid out above, CORRECT-ANNOTATOR provides a proof-of-concept for text-transformation based annotation. For these reasons, it would be a unique supplement to the CLARIN infrastructure and useful for researchers in the CLARIN community, especially to those within the field of learner corpora.

2.3.1 Current Challenges And Future Work

The are, however, some areas that might prove to be difficult in further developing and extending CORRECT-ANNOTATOR. The first difficulty is in how the heuristic for identifying the language errors is implemented. Since each language error is bound to specific user events, one has to be well versed in JavaScript as well as its event-driven semantics, in order to implement a new language error type.

Furthermore, choosing the corpus text to be editable comes with another set of implementation difficulties. Corrections made in the text always have to be matched to the original text and user edit operations have to be meticulously guarded against, in order to protect the integrity of the annotation. This is especially the case when it comes to corrections on a sentence level, where a user might make a series of edits in order to annotate just one language error. The more levels of edits a user makes for one language error, the bigger the risk for false positioning - for example a user might, after all, for one language error, cut, copy, paste one word of the original text and insert a non-existent word. These are obviously less prevalent difficulties in a traditional annotation tool, because of the fact that user operations are limited to the creation of external data.

At the time of writing this paper, CORRECT-ANNOTATOR has yet to be tested in an annotation project. Beside a test set of defined error types, the comprehensive language error taxonomy of SweLL has not yet been defined. Before the program would be used for actual annotation, the program would have to be tested in a production environment. That means using the program to annotate real second language learner texts with a real language error taxonomy and evaluating the results and usability of the program. Only then can it be determined if CORRECT-ANNOTATOR, would be a valuable tool for annotating the future SweLL L2-corpus.

3 Acknowledgements

This work has been partly supported by an infrastructure grant from the Swedish Research Council (SWE-CLARIN, project 821-2013-2003).

References

Jeremy Ashkenas. 2010. Backbone.js online documentation version 1.3.3. http://backbonejs.org/. Accessed: 2016-05-11.

Margareta Westergren Axelsson. 2000. Use-the uppsala student english corpus: an instrument for needs analysis. *ICAME journal*, 24:155–157.

Richard Eckart de Castilho, Chris Biemann, Iryna Gurevych, and Seid Muhie Yimam. 2014. Webanno: a flexible, web-based annotation tool for clarin. In *Proceedings of the CLARIN Annual Conference (CAC)*.

Iria del Río, Sandra Antunes, Amália Mendes, and Maarten Janssen. 2016. Towards error annotation in a learner corpus of portuguese. In *Proceedings of the joint workshop on NLP for Computer Assisted Language Learning and NLP for Language Acquisition at SLTC, Umeå, 16th November 2016*, pages 8–17. Linköping University Electronic Press.

Sylviane Granger, Estelle Dagneaux, Fanny Meunier, and Magali Paquot. 2002. *International corpus of learner English*. Presses universitaires de Louvain.

Sylviane Granger. 2003. Error-tagged learner corpora and call: A promising synergy. *CALICO journal*, pages 465–480.

Björn Hammarberg. 2013. Andraspråksforskning med asu-korpusen. *Nordand: nordisk tidsskrift for andrespråksforskning*, 8(1):7–33.

Erhard Hinrichs and Steven Krauwer. 2014. The CLARIN Research Infrastructure: Resources and Tools for e-Humanities Scholars. *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*, pages 1525–1531, May.

Viggo Kann, Rickard Domeij, Joachim Hollman, and Mikael Tillenius. 2001. Implementation aspects and applications of a spelling correction algorithm. *Text as a Linguistic Paradigm: Levels, Constituents, Constructs. Festschrift in honour of Ludek Hrebicek*, 60:108–123.

Helen Langone, Benjamin R Haskell, and George A Miller. 2004. Annotating wordnet. Technical report, DTIC Document.

Cristóbal Lozano. 2009. Cedel2: Corpus escrito del español l2.

John Resig. 2017. jquery online documentation version 1.9. http://api.jquery.com/. Accessed: 2017-03-16.

Peter Siemen, Anke Lüdeling, and Frank Henrik Müller. 2006. Falko-ein fehlerannotiertes lernerkorpus des deutschen. *Proceedings of Konvens 2006*.

John McHardy Sinclair. 2004. *How to use corpora in language teaching*, volume 12. John Benjamins Publishing.

Han Sloetjes and Peter Wittenburg. 2008. Annotation by category: Elan and iso dcr. In *LREC*.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107. Association for Computational Linguistics.

Elena Volodina, Beata Megyesi, Mats Wirén, Lena Granstedt, Julia Prentice, Monica Reichenberg, and Gunlög Sundberg. 2016. A friend in need? research agenda for electronic second language infrastructure. *Proceedings of SLTC*.