Error Coding of Second-Language Learner Texts Based on Mostly Automatic Alignment of Parallel Corpora

Dan Rosén
Språkbanken, Department of Swedish
University of Gothenburg, Sweden
dan.rosen@svenska.gu.se

Mats Wirén
Department of Linguistics
Stockholm University, Sweden
mats.wiren@ling.su.se

Elena Volodina
Språkbanken, Department of Swedish
University of Gothenburg, Sweden
elena.volodina@svenska.gu.se

Abstract

Error coding of second-language learner text, that is, detecting, correcting and annotating errors, is a cumbersome task which in turn requires interpretation of the text to decide what the errors are. This paper describes a system with which the annotator corrects the learner text by editing it prior to the actual error annotation. During the editing, the system automatically generates a parallel corpus of the learner and corrected texts. Based on this, the work of the annotator consists of three independent tasks that are otherwise often conflated: correcting the learner text, repairing inconsistent alignments, and performing the actual error annotation.

1 Introduction

Error coding is a highly useful way of increasing the value of second-language learner data, but it is also ”beset with difficulties” (Granger, 2008, page 266). Like all manual annotation, it is very time-consuming, but the problem is exacerbated by the fact that the text must be interpreted to decide what the correct forms (the target hypotheses) are. Our approach is based on regarding the learner source text and the corrected text as a parallel corpus. The annotator makes the corrections of the learner text using a text editor upon which the words of the two resulting texts are automatically aligned. The annotator may re-align inconsistent links, and can then perform the actual error annotation.

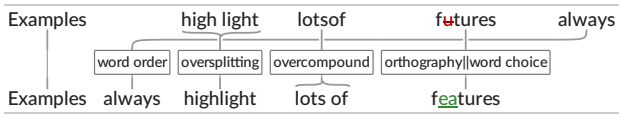
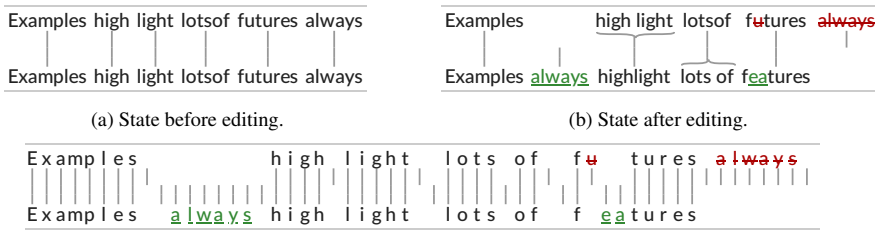


Figure 1: Example learner sentence (above), corrected target hypotheses (below) and error annotation.

Figure 1 shows a hypothetical learner sentence, its correction and the automatic alignments generated by the system. The misspelled or wrongly chosen word *futures* is aligned with the corrected word *features*. Movements are kept track of by alignments of the words involved, for example, *always* at the end of the learner sentence with *always* at the second position in the corrected sentence. Compound oversplitting, as in the linking of *high light* and *highlight*, is represented by a many-to-one alignment. Conversely, overcompounding, as in *lotsof* and *lots of*, is represented by a one-to-many alignment.

2 Related work

Error coding in second-language learner texts is typically made with a purpose-built editing tool, using a hierarchical set of error categories (Granger, 2008), for example, the XML editor Oxygen in the ASK



(a) State before editing.

(b) State after editing.

(c) Character based diff: this is lifted to the token level in Figure 2b. (not shown to the user)

Figure 2: Before and after editing. The automatic aligner gets most words correct and the user will later manually connect the unlinked words.

project (Tenfjord et al., 2006), the TEITOK editor in the COPLE2 project (Mendes et al., 2016), and QAWI in the QALB project (Obeid et al., 2013). In many systems, however, the conceptually separate stages of error coding — error detection, correction and annotation (Ellis, 1994; Granger, 2008, page 266) — are conflated such that correction is carried out as a subtask of error annotation, for example, Tenfjord et al. (2006) and Mendes et al. (2016). In contrast, two projects that factor out correction as an independent step are MERLIN (Boyd et al., 2014) and CzeSL (Hana et al., 2012). The latter of these, with its parallel corpus editor *feat*, is the one most closely related to our system. The main difference is that our editor makes word alignments automatically. On the other hand, *feat* includes two tiers of target hypotheses, roughly corresponding to correction of spelling and grammatical errors, respectively, whereas our system currently only has one level of correction.

In translation and related fields, several tools for manual or semi-automatic word alignment of parallel corpora have been developed, for example, *Interactive Linker* (Merkel et al., 2003), *Yawat* (Germann, 2008), and, for multiparallel corpora, *Hierarchical Alignment Tool* (Graën, 2018). An assumption in our system is that the differences between the source and target texts are relatively small, which makes it possible for an automatic aligner to rely only on orthographic differences. Such differences, for example, the longest common subsequence ratio by Melamed (1999), have been used for alignment of parallel corpora, but not for alignment of learner corpora as far as we are aware of.

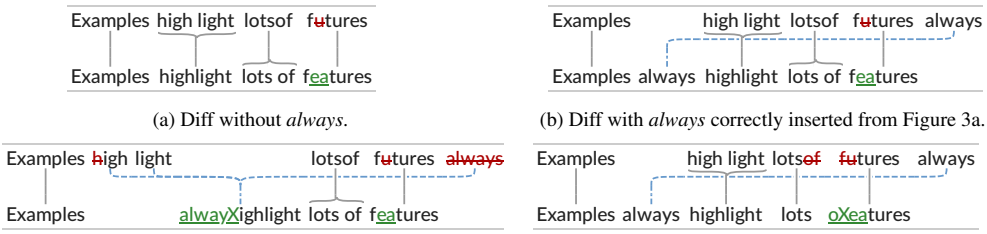
3 Correction and alignment

3.1 Basic workings

The system interface includes two main panes: one editable text pane displaying the text being corrected, and a graphic rendering of the links between the source text and the corrected text. An optional third pane displays the static learner text. Initially the corrected text is identical to the source text, and each word is aligned to its copy. In the example "*Examples high light lotsof futures always*", the graph initially looks as in Figure 2a. To correct a text, the annotator uses the editable text pane, which works like a standard display editor. Editing operations can be made in any order, and upon each change, the system immediately updates the alignments between the two texts. Changing the text to "*Examples always highlight lots of features*" results in the alignment in Figure 2b.

Our system builds a corrected version aligned with the learner text using an underlying JSON representation of the tokens, the groups of links between these, and any labels attached to the groups. How is this calculated? We start with a standard diff edit script on the *character level*. Internally, the representation in Figure 2c is generated, which is calculated using Myers' diff algorithm (Myers, 1986) provided by the diff-match-patch[1] library. Each character is associated with the token it originates from. Next, these character-level alignments are lifted to the token level. Spaces are not used for alignment to avoid giving rise to too many false positives. We can now read off from this representation which tokens should be aligned. For each pair of matching characters, we add a link to their corresponding tokens. For example,

[1] `https://github.com/google/diff-match-patch`



(a) Diff without *always*.

(b) Diff with *always* correctly inserted from Figure 3a.

(c) Editing from Figure 3b between *always* and *highlight*.

(d) Editing from Figure 3b between *of* and *features*.

Figure 3: Aligning and editing in the presence of a manually aligned word (here *always*).

since there is a link between the *h* in *high* and *highlight*, these two words are aligned. Furthermore, since there is a link between the *l* in *light* to this target word, all these three words should also be aligned. There are no other characters linked to characters from these tokens, so exactly these three will become a group. The other tokens are connected analogously.

3.2 Manual alignments: word order changes and inconsistent links

In Figure 2b, the two occurrences of the word *always* are not aligned. The user can correct this error by selecting both occurrences of *always* and clicking the *group* button (not shown here). After this grouping we are in a state where the parallel structure has one manual alignment pertaining to the word *always*, with all other words being candidates for automatic (re-)alignment. To (re-)align these we carry out the same procedure as before but excluding the manually aligned *always*: We first *remove* manually aligned words, align the rest of the text automatically (see Figure 3a), and then *insert* the manually aligned words again in their correct position (Figure 3b). Here the correct position is where they were removed from the respective texts. The editor indicates that this link is manually aligned by colouring it blue and (for the purposes of this article) making it dotted. These links interact differently with automatically aligned (grey) links. How this works is explained in the next section.

3.3 Editing in the presence of both manual and automatic alignments

The tokens that have been manually aligned are remembered by the editor. The user may now go on editing the target hypothesis. Things are straightforward as long as the edit takes place wholly in either an automatically aligned passage or a manually aligned passage. When editing across these boundaries, the manual segment is contagious in the sense that it extends as much as needed. For example, if we select *always highlight* in Figure 3b and replace it with *alwayXighligt*, the state becomes as shown in Figure 3c. However, if we cross the boundary of *of features* in the same starting state of Figure 3b to *oXeatures*, we get the state of Figure 3d. Here the edit is not contagious: the automatic alignment decided not to make a big component and instead chose to split to align the words independently. In case the manual aligner has absorbed too much material, our editor provides a way of removing the manual alignment tag. The editor will then fall back to the automatic aligner for those tokens.

4 Error annotation

Error categories can be seen as relations between words in the learner text and corrected text, and are therefore associated with the alignments, as in Figure 1. Our error taxonomy is inspired by that of ASK (Tenfjord et al., 2006), with two hierarchical levels (main categories and subcategories).[2] Error annotation is carried out by selecting one or several words in the learner and corrected texts, or by selecting the corresponding link(s). A pop-up menu is displayed and the annotator selects one or more error categories, whereupon the system attaches these categories to the corresponding links as shown in Figure 1. For meaningful error annotation to take place, some correction must have been made, but the order between the tasks is otherwise arbitrary.

[2]The names of the categories differ from the example labels used in Figure 1.

5 Concluding remarks

CLARIN currently lacks an established infrastructure for research in second-language learning. The work reported here is part of the SweLL project aiming at filling this gap for Swedish, but our intent is to make the methods as generally applicable as possible. To this end, the system described here is free software under the MIT license.[3] Also, the error taxonomy is customisable in the system. We expect to soon be able to provide a more complete description of the system and the rationale for our methodology, including the error taxonomy, practical experiences with annotators, use of the system for anonymisation, and functionalities for management of annotators.

Acknowledgements

This work has been supported by Riksbankens Jubileumsfond through the SweLL project (grant IN16-0464:1, `https://spraakbanken.gu.se/eng/swell_infra`). Some of the basic ideas were first tested in a pilot system by Hultin (2017), supervised by Robert Östling and Mats Wirén. We are grateful for valuable comments and feedback on the system from Markus Forsberg, Lars Borin and our co-workers in the SweLL project.

References

[Boyd et al.2014] Adriane Boyd, Jirka Hana, Lionel Nicolas, Detmar Meurers, Katrin Wisniewski, Andrea Abel, Karin Schöne, Barbora Štindlová, and Chiara Vettori. 2014. The MERLIN corpus: Learner Language and the CEFR. In *LREC'14*, Reykjavik, Iceland. European Language Resources Association (ELRA).

[Ellis1994] Rod Ellis. 1994. *The Study of Second Language Acquisition*. Oxford University Press, Oxford.

[Germann2008] Ulrich Germann. 2008. Yawat: Yet another word alignment tool. In *Proc. ACL: HLT Demo Session*, pages 20–23. Association for Computational Linguistics.

[Granger2008] Sylviane Granger. 2008. Learner corpora. In Anke Lüdeling and Merja Kytö, editors, *Corpus Linguistics. An International Handbook*, volume 1, chapter 15, pages 259–275. Mouton de Gruyter, Berlin.

[Graën2018] Johannes Graën. 2018. *Exploiting Alignment in Multiparallel Corpora for Applications in Linguistics and Language Learning*. Ph.D. thesis.

[Hana et al.2012] Jirka Hana, Alexandr Rosen, Barbora Štindlová, and Petr Jäger. 2012. Building a learner corpus. In *LREC'12*, Istanbul, Turkey, may. European Language Resources Association (ELRA).

[Hultin2017] Felix Hultin. 2017. Correct-Annotator: An Annotation Tool for Learner Corpora. CLARIN Annual Conference 2017, Budapest, Hungary.

[Melamed1999] I. Dan Melamed. 1999. Bitext maps and alignment via pattern recognition. *Computational Linguistics*, 25(1):107–130, March.

[Mendes et al.2016] Amália Mendes, Sandra Antunes, Maarten Janssen, and Anabela Gonçalves. 2016. The COPLE2 corpus: a learner corpus for Portuguese. In *LREC'16*.

[Merkel et al.2003] Magnus Merkel, Michael Petterstedt, and Lars Ahrenberg. 2003. Interactive word alignment for corpus linguistics. In *Proc. Corpus Linguistics 2003*.

[Myers1986] Eugene W. Myers. 1986. An O(ND) difference algorithm and its variations. *Algorithmica*, 1(1):251–266.

[Obeid et al.2013] Ossama Obeid, Wajdi Zaghouani, Behrang Mohit, Nizar Habash, Kemal Oflazer, and Nadi Tomeh. 2013. A Web-based Annotation Framework For Large-Scale Text Correction. In *The Companion Volume of the Proceedings of IJCNLP 2013: System Demonstrations*, pages 1–4. Asian Federation of Natural Language Processing.

[Tenfjord et al.2006] Kari Tenfjord, Paul Meurer, and Knut Hofland. 2006. The ASK corpus: A language learner corpus of Norwegian as a second language. In *LREC'06*, pages 1821–1824.

[3] `https://github.com/spraakbanken/swell-editor`.