WAMP: Writing, Annotation, and Marking Platform

Geonsik Moon[*], **Muhammad Reza Qorib**[*], **Daniel Dahlmeier**[†] and **Hwee Tou Ng**[*]
[*] Department of Computer Science, National University of Singapore
[†] SAP Innovation Center Singapore
moon97@nus.edu.sg, mrqorib@comp.nus.edu.sg, nght@comp.nus.edu.sg
d.dahlmeier@sap.com

Abstract

Creating annotated corpora has always been one of the major challenges for grammatical error correction (GEC). Annotated corpora are critical since they enable us to evaluate the accuracy of GEC systems by providing gold-standard references. In this paper, we propose a web-based annotation tool – Writing, Annotation, and Marking Platform (WAMP) – that tackles this issue of generating annotated corpora by allowing annotators to annotate essays with ease and export the resulting annotated essays for use in GEC research. The source code and a demo video of WAMP are publicly available at: https://github.com/nusnlp/wamp.

1 Introduction

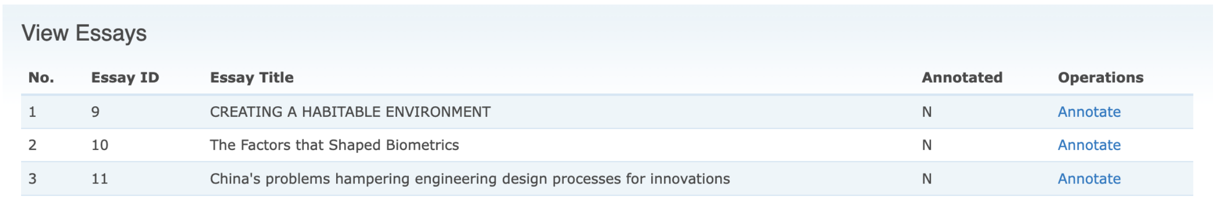
Grammar is an important part of language learning, especially in writing. Incorrect grammar could misrepresent some of the meanings that a writer is trying to convey and damage the credibility of the writer, ultimately resulting in ineffective communication. Despite having learned English in primary and secondary education, non-native English speakers can still struggle with grammatical errors in writing. Subject-verb agreement and verb tense are among the most frequent error types (Singh et al., 2017). As such, grammatical error correction (GEC), formulated as the task of automatically detecting and correcting grammatical errors in a text (Chollampatt et al., 2016; Chollampatt and Ng, 2018; Qorib et al., 2022; Bryant et al., 2023), has gained attention over the past decade as a useful application for language learners. GEC facilitates faster language learning by providing a better alternative for these students to check their writing, replacing manual checking.

However, one of the biggest challenges for GEC arises from generating high-quality annotated corpora. Due to its time-consuming nature, manual annotation is still regarded as the major bottleneck for building annotated corpora that are large enough to train and test systems for many NLP tasks (Neves and Ševa, 2019). GEC is no exception, as it also requires a gold-standard reference in order to evaluate the accuracy of GEC systems and analyze the specific areas in which these systems under-perform. The NUS Corpus of Learner English (NUCLE) (Dahlmeier et al., 2013), a large annotated corpus of learner essays, was created in collaboration with the Centre for English Language Communication (CELC) at NUS. Comprising about 1,400 essays written by undergraduate students at NUS, NUCLE contains a total of over one million words which were annotated with error tags and corrections by a team of professional English language instructors at NUS (Dahlmeier et al., 2013). In building the NUCLE corpus, a web-based annotation tool – Writing, Annotation, and Marking Platform (WAMP) – was utilized to allow the instructors to annotate student essays online, using a web browser of their choice.

To facilitate the creation of additional annotated corpora for GEC, we provide the system overview, functionalities, and system implementation of WAMP in this paper. We also introduce two new enhancements to the original WAMP that allow greater usability of the tool for GEC research: (1) The user can choose from different sets of error tags that are available for annotators. (2) The user can export each annotated essay into the M2 file format, which displays the start word index, end word index, and the error type of each edit. This M2 file can then be used as a gold-standard reference for the MaxMatch (M^2) scorer (Dahlmeier and Ng, 2012) and the ERRANT (Bryant et al., 2017) scorer to evaluate GEC systems.

2 System Overview

Writing, Annotation, and Marking Platform (WAMP) is a web-based annotation tool, built us-


View Essays

No.	Essay ID	Essay Title	Annotated	Operations
1	9	CREATING A HABITABLE ENVIRONMENT	N	Annotate
2	10	The Factors that Shaped Biometrics	N	Annotate
3	11	China's problems hampering engineering design processes for innovations	N	Annotate

Figure 1: The web interface in WAMP for viewing essays.

ing Drupal (version 6.13)[1], a PHP-based content management system. Drupal can be easily customized to extend the functionalities of the software with the use of modules. Other than the community-contributed modules (Appendix A.1) that are shipped together with the Drupal release, a custom module named "WAMP", consisting of two sub-modules "Essay" and "Administration", extends the main functionalities of WAMP.

2.1 WAMP Module

The WAMP module is the parent module that encompasses the two aforementioned sub-modules, and these sub-modules inherit all the functions of the WAMP module. Other than the administrator who manages WAMP, the main user of the software is the "annotator" and the WAMP module specifically contains the navigational user interface for the annotator to view and annotate essays, as illustrated in Figure 1. The WAMP module also allows the administrator to toggle between the mode production or testing (for debugging purpose).

2.2 Essay Sub-Module

The Essay sub-module comprises essay-related management functionalities for the administrator to view all the annotations created by the annotators. Additionally, the administrator can check if an essay has been flagged to be of poor quality or invalid format, and delete the essay to make it unavailable to annotators accordingly. The functionalities for annotating essays are also contained under the Essay sub-module, which will be discussed in more detail under the Annotate Essays subsection.

2.3 Administration Sub-Module

The Administration sub-module is responsible for extending data management functionalities for the administrator, which mainly consists of importing and exporting files. Essays are imported into WAMP in XML (Extensible Markup Language)[2] format. WAMP further processes the imported file to extract the essay title and body, and insert the relevant information into the database. The functionalities for importing essays will be discussed in more detail under the Import Essays subsection. WAMP supports two types of data export: (1) XML and (2) M2. The administrator can export the entire annotation data from the WAMP database into a single XML file. This exported XML file contains the data for all essays and the relevant annotations for all grammatical errors identified and corrected by the annotators. WAMP also allows export of data in M2 file format. Unlike XML export, M2 export is executed for each individual annotator, i.e., the annotation data for a single essay by a single annotator is exported. The exported M2 file contains all the sentences of an essay in separate lines. Each sentence is followed by lines that indicate the start and end word index of the sequence of words identified as an error in the sentence, and the appropriate error type provided by the annotator. Exporting essays will be discussed in greater detail under the Export Essays subsection.

Additionally, under the Administration sub-module, the administrator can also select between two different sets of error tags to be made available to the annotators. Currently, WAMP supports NUCLE (Dahlmeier et al., 2013) as the default set of error tags, with an additional support for the set of ERRANT (Bryant et al., 2017) error tags. NUCLE introduced an extensive list of 28 error tags, ranging from frequently found error types such as verb tense error and subject-verb agreement error, to more niche error types, such as poor citation practice and unclear meaning. Similarly, (Bryant et al., 2017) proposed another error tag scheme that consists of 25 main error types. WAMP is designed in a way that allows easy extension to new sets of error tags. The customization only requires the addition of a new HTML file that contains the

[1]`https://www.drupal.org/project/drupal/releases/6.13`

[2]`https://developer.mozilla.org/en-US/docs/Web/XML/XML_introduction`

new error tags and a simple modification to the Administration sub-module. As such, WAMP can be easily customized to accommodate the needs of other GEC research that requires annotated corpora with grammatical errors categorized by a different set of error tags.

3 Functionalities

3.1 Import Essays

Before an annotator can start the annotation on the essays, the administrator first needs to import the essays into WAMP. Each essay has two main components, which are the title and the main body. To import an essay into WAMP, it needs to be in XML format, as illustrated in Figure 2. It is also possible to import multiple essays into WAMP with a single import XML file. Once the XML file is loaded from the database after the upload, the essay is inserted into the database through post-processing of the import XML file. The annotators can now view the list of uploaded essays, as shown in Figure 1, and click on "Annotate" to begin the annotation process.

```
<instance title="CREATING␣A␣HABITABLE␣
    ENVIRONMENT">
    <context>
        Humans have many basic needs and
             one of them is to have an
            environment that can...
        Some countries are having
            difficulties in managing a
            place to live for their
            citizen as they...
    </context>
</instance>
```

Figure 2: An example illustrating the import XML format. The details are explained in Appendix A.2.

3.2 Annotate Essays

Any annotator account can be used to annotate an essay. The following steps to annotate essays are based on the assumptions that (1) an essay has been imported to the database and (2) an annotator account has been created.

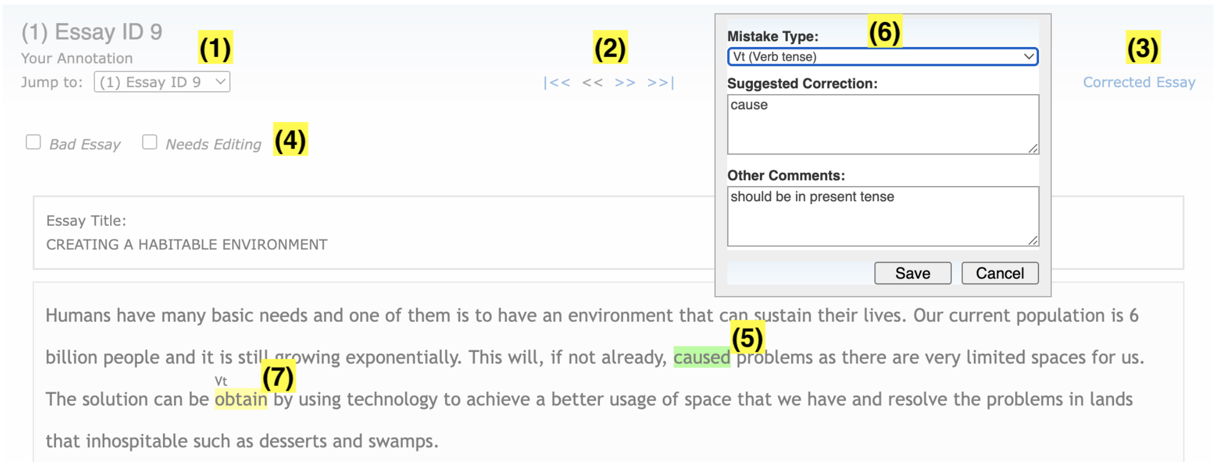
An annotator can utilize the WAMP interface to perform annotation on an uploaded essay. However, before the annotator begins to annotate an essay, WAMP allows the annotator to flag the essay to be of poor quality or invalid format. More specifically, if the quality of the essay is very poor and annotations should not be made on it, the annotator can notify the administrator by checking the "Bad Essay" check box. Similarly, if the essay is truncated, poorly formatted, or contains unexpected Unicode characters, the annotator can check the "Needs Editing" check box to notify the administrator to modify the essay before anyone can proceed to annotate the essay. These check boxes can be found in label (4) of Figure 3. The information is shared among all annotators. Hence, if an essay has been flagged as either "Bad Essay" or "Needs Editing", all annotators who have access to the essay will be notified with a warning whenever they load the flagged essay.

3.2.1 Adding Annotations

Once an annotator confirms that there is no issue with an essay, the annotator can proceed with the annotation. The action of annotating an essay can be broken down into three main steps, which are identify, classify, and correct. As shown in label (5) of Figure 3, the annotator can use the cursor to highlight an arbitrary contiguous text span (containing multiple words in general) and identify the grammatical error. Once an error has been highlighted, a pop-up box prompts the annotator to choose the appropriate error type, as illustrated in label (6) of Figure 3. The available error types are based on the predefined set of error tags, configured in the aforementioned Administration sub-module. Lastly, the annotation is completed when the annotator enters an appropriate correction string to correct the error and clicks "Save." Once an annotation is completed, the identified error will be highlighted in yellow with the assigned error tag on top, as indicated in label (7) of Figure 3. Optionally, WAMP allows the annotator to add a comment to clarify the annotation.

3.2.2 Modifying Annotations

WAMP also allows an annotator to modify or delete an existing annotation. After an annotation is saved, clicking on the highlighted text span (as illustrated in label (5) of Figure 3) opens a pop-up window, as shown in Figure 4. The annotator can choose a different error tag, enter another correction string or comment for that annotation by clicking on the "Edit" button. If the annotator considers an existing annotation to be unnecessary, the annotator can simply click on the "Delete" button to delete the annotation from the essay. A sample annotated essay is shown in Figure 5.


(1) Essay ID 9

Your Annotation

Jump to: (1) Essay ID 9

|<< << >> >>|

Corrected Essay

Bad Essay Needs Editing

Mistake Type: Vt (Verb tense)

Suggested Correction: cause

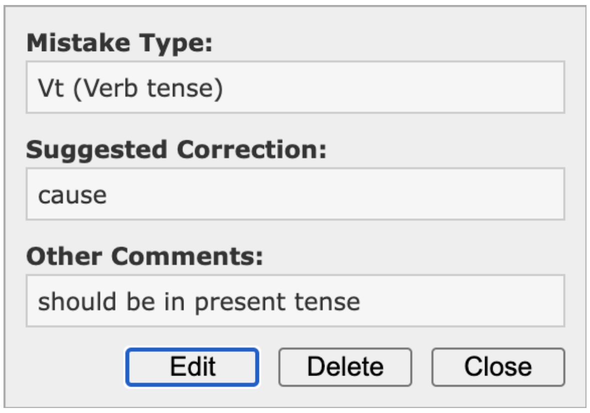
Other Comments: should be in present tense

Save Cancel

Essay Title:
CREATING A HABITABLE ENVIRONMENT

Humans have many basic needs and one of them is to have an environment that can sustain their lives. Our current population is 6 billion people and it is still growing exponentially. This will, if not already, caused problems as there are very limited spaces for us. The solution can be obtain by using technology to achieve a better usage of space that we have and resolve the problems in lands that inhospitable such as desserts and swamps.

Figure 3: The web interface in WAMP for annotating an essay.


Mistake Type: Vt (Verb tense)

Suggested Correction: cause

Other Comments: should be in present tense

Figure 4: An example illustrating a pop-up menu for annotation.

3.2.3 Characteristics of WAMP

One characteristic of WAMP is that the start (or end) position of an erroneous text span can be the location of any character in an essay, and does not need to be the start (or end) of a word. By allowing annotation in fine granularity and imposing minimal constraints, WAMP provides an environment that closely resembles annotating with pen and paper for the annotators. Additionally, WAMP ensures that no annotation data is lost or damaged during the annotation process by saving changes after every annotation operation. This can prove useful in cases of unexpected connection or hardware failures.

Although WAMP allows multiple annotators to collaborate on annotating the same essay by sharing an annotator account, this workflow is not typical in the context of constructing a GEC corpus. For GEC annotation, it is preferable to have as many different annotations as possible for the same text as there are usually multiple possible ways of correcting an erroneous text (Bryant and Ng, 2015).

Lastly, WAMP allows an annotator to view the corrected version of an annotated essay, which is available as a link in label (3) of Figure 3. The corrected essay simply replaces the existing erroneous text spans with the proposed corrections, displaying each corrected text span in bold and enclosing it within a box. This functionality facilitates the revision of essays for annotators by making it possible to view a corrected essay after the proposed corrections are applied, without having to check each annotation individually. As such, WAMP is unique in that it is an annotation platform that offers an easy interface to highlight errors, classify error types, and view corrected essays.

3.3 Export Essays

Once all the annotation tasks are completed, the administrator can export the annotation data in XML format, as shown in Figure 6. The exported XML file contains all the essays and the annotations that have been saved in the WAMP database so far. WAMP facilitates the export process by allowing extraction of the entire data into a single XML file. The export XML file contains the list of essays that have been uploaded to WAMP. Within each essay are the annotations, if any, and each annotation is identified by the annotator's name to distinguish annotations on the same essay by different annotators.

As previously mentioned, WAMP also supports export of annotation data in M2 format. There are several scorers available to evaluate the performance of GEC systems on a data set and two of the most widely utilized are the MaxMatch

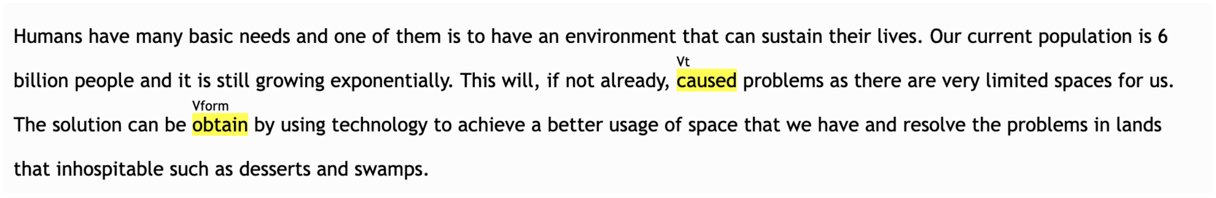

Humans have many basic needs and one of them is to have an environment that can sustain their lives. Our current population is 6 billion people and it is still growing exponentially. This will, if not already, caused problems as there are very limited spaces for us. The solution can be obtain by using technology to achieve a better usage of space that we have and resolve the problems in lands that inhospitable such as desserts and swamps.

Figure 5: An example illustrating an annotated essay.

```
<annotation annotator="John␣Smith">
    <mistake start="/0.210" end="/0.216"
        >
        <type>Vt</type>
        <correction>cause</correction>
        <comments>should be in present
            tense</comments>
    </mistake>
    <mistake start="/0.287" end="/0.293"
        >
        <type>Vform</type>
        <correction>obtained</correction
            >
        <comments/>
    </mistake>
</annotation>
```

Figure 6: An example illustrating the export XML format. The details are explained in Appendix A.3.

(M^2) scorer (Dahlmeier and Ng, 2012) and the ERRANT (Bryant et al., 2017) scorer. Used as the official GEC scorer for the CoNLL-2013 (Ng et al., 2013) and CoNLL-2014 (Ng et al., 2014) shared tasks, the MaxMatch (M^2) scorer computes the F-score over the optimal phrase-level alignment between a source sentence and a system hypothesis that achieves the highest overlap with the gold-standard annotation (Dahlmeier and Ng, 2012; Náplava et al., 2022). It was also used for other non-English corpora, including the Falko-MERLIN corpus (Boyd, 2018) in German, and the RULEC-GEC corpus (Rozovskaya and Roth, 2019) in Russian. Both the MaxMatch (M^2) scorer and the ERRANT scorer utilize M2 files for evaluation. As shown in Figure 7, the exported M2 file contains a list of tokenized original sentences, preceded by S. Each of these sentences is optionally followed by lines, preceded by A. Each line contains information about an annotation on the associated sentence, including the start and end token offset, the error type, and the tokenized correction string. This indicates that WAMP not only facilitates seamless annotation on essays but also generates annotated gold-standard references that can be further utilized directly for evaluation of GEC system performance.

4 System Implementation

WAMP is a web-based application, built using the LAMP[3] full-stack framework. The specific dependencies are listed in Appendix A.4. To facilitate the deployment of WAMP by standardizing the development environment, we use Docker (version 4.13.1)[4] to set up the required LAMP stack. We build our Docker based on the Docker image[5] of Ubuntu 9.10. After running the Docker image as a standalone container, the source code of WAMP is mounted to the container. One of the benefits of Docker is that other users can easily replicate the environment setup on any device and run the software without having to worry about dependency issues. We believe this is critical for the usability of WAMP and any other annotation tools that serve a similar purpose.

5 Related Work

Grammatical error correction is an active area of research in natural language processing (Chollampatt et al., 2016; Chollampatt and Ng, 2018; Qorib et al., 2022; Bryant et al., 2023). Both commercial (e.g., Grammarly) and open-source GEC tools (e.g., ALLECS (Qorib et al., 2023)) are available. In tandem, the need for new GEC corpora also increases to support GEC development with more training and evaluation data. Unfortunately, as of now, there is no publicly available open-source GEC annotation tool to help with the creation of GEC corpora. Andersen (2011) reported that annotating GEC corpora with a graphical tool can significantly increase the annotation speed compared to using a text editor. The author made a simple annotation tool for the experiment, but the tool is not explained in detail and not publicly available.

One non-free tool that can be used for GEC anno-

[3]https://www.ibm.com/cloud/learn/lamp-stack-explained

[4]https://docs.docker.com/desktop/release-notes/

[5]https://github.com/iComputer7/ancient-ubuntu-docker

```
S This will , if not already , caused problems as there are very limited spaces for us .
A 7 8|||Vt|||cause|||REQUIRED|||-NONE-|||0

S The solution can be obtain by using technology to achieve a better usage of space that we
have and resolve the problems in lands that inhospitable such as desserts and swamps .
A 4 5|||Vform|||obtained|||REQUIRED|||-NONE-|||0
```

Figure 7: An example illustrating the exported M2 format.

tation is Write&Improve[6], which is an online platform to assist English learners to improve their writing (Yannakoudakis et al., 2018). Write&Improve allows English learners to submit their essays and get feedback from an automated system or a teacher. In our experience of using Write&Improve, we could not find a way of labeling error type when annotating a text. Furthermore, the system is unable to export the list of corrections to be applied to a text, which is important for the GEC task. This is because the system was designed to focus more on tracking students' learning progress instead of creating a GEC corpus. The lack of error-type labeling and correction export features is also the reason why text rewriting tools (Xu et al., 2019; Goldfarb-Tarrant et al., 2019) are not suitable for GEC annotation.

Kutuzov and Kuzmenko (2015) designed a tool for grammatical error detection (GED) annotation by integrating BRAT (Stenetorp et al., 2012) with a linguistic analyzer (FreeLing (Padró and Stanilovsky, 2012)) and a spellchecker (Aspell[7]). BRAT is a multi-task text annotation tool that can be used for part-of-speech tagging, named entity recognition, dependency parsing, and other NLP tasks. BRAT has an intuitive interface similar to WAMP, but BRAT is not suitable for GEC corpora annotation. This is because for each text span, one can only assign a label from a pre-determined list of labels instead of providing a free-form correction string. This is also the reason why other sequence-tagging annotation tools (Yang et al., 2018; Zhang et al., 2021) are not suitable for GEC annotation, which requires associating a text span with a free-form correction string and a label for its error type.

Machine translation post-editing tools (Lee et al., 2021) or text simplification annotation tools (Stodden and Kallmeyer, 2022) are also not suitable for GEC annotation, because the tools expect a pair of sentences for each data instance. This means that the annotator needs to rewrite the whole text in full first, then aligns the words in both sentences. This will result in excessive rewriting and a slower annotation process, compared to manual annotation with a text editor.

In summary, GEC annotation needs a specialized tool that cannot be easily substituted by other natural language processing annotation tools.

6 Conclusion

In this paper, we introduce WAMP, a web-based annotation platform to facilitate the process of manual annotation and export the annotation data in XML and M2 format. WAMP has already proven its utility in building the NUCLE corpus, a large annotated corpus that contains over one million words which are completely annotated by professional English instructors using WAMP (Dahlmeier et al., 2013). With the new enhancements in place, WAMP brings even greater values in generating annotations for annotated corpora by offering more flexibility in defining the error tags and converting the annotation data into the M2 file format that can be used as references for evaluating GEC systems.

For future work, WAMP can be extended to include learning analytics features, such as keeping track of the counts of different error types made by different groups of learners. This will facilitate the analysis of the error profile of different learners and enable personalized and targeted assistance to learners.

Acknowledgements

We thank Jun Hong Pua and Hong Guan Chan for implementing WAMP. This research is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-RP-2019-014).

Limitations

As an annotation tool, WAMP is tailored for the text correction task, by allowing a user to easily highlight any span of text and adding the corrected text as replacement. As such, WAMP is less suitable as an annotation tool for other natural language

[6] https://writeandimprove.com/
[7] http://aspell.net/

processing tasks such as named entity recognition, coreference resolution, etc.

References

Øistein E. Andersen. 2011. Semi-automatic ESOL error annotation. *English Profile Journal*, 2:e1.

Adriane Boyd. 2018. Using Wikipedia edits in low resource grammatical error correction. In *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*, pages 79–84, Brussels, Belgium. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada. Association for Computational Linguistics.

Christopher Bryant and Hwee Tou Ng. 2015. How far are we from fully automatic high quality grammatical error correction? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 697–707, Beijing, China. Association for Computational Linguistics.

Christopher Bryant, Zheng Yuan, Muhammad Reza Qorib, Hannan Cao, Hwee Tou Ng, and Ted Briscoe. 2023. Grammatical error correction: A survey of the state of the art. *Computational Linguistics*, 49(3).

Shamil Chollampatt and Hwee Tou Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5755–5762.

Shamil Chollampatt, Kaveh Taghipour, and Hwee Tou Ng. 2016. Neural network translation models for grammatical error correction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2768–2774.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572, Montréal, Canada. Association for Computational Linguistics.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner English: The NUS corpus of learner English. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31, Atlanta, Georgia. Association for Computational Linguistics.

Seraphina Goldfarb-Tarrant, Haining Feng, and Nanyun Peng. 2019. Plan, write, and revise: an interactive system for open-domain story generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 89–97, Minneapolis, Minnesota. Association for Computational Linguistics.

Andrey Kutuzov and Elizaveta Kuzmenko. 2015. Semi-automated typical error annotation for learner English essays: integrating frameworks. In *Proceedings of the fourth workshop on NLP for computer-assisted language learning*, pages 35–41, Vilnius, Lithuania. LiU Electronic Press.

Dongjun Lee, Junhyeong Ahn, Heesoo Park, and Jaemin Jo. 2021. IntelliCAT: Intelligent machine translation post-editing with quality estimation and translation suggestion. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 11–19, Online. Association for Computational Linguistics.

Jakub Náplava, Milan Straka, Jana Straková, and Alexandr Rosen. 2022. Czech grammar error correction with a large and diverse corpus. *Transactions of the Association for Computational Linguistics*, 10:452–467.

Mariana Neves and Jurica Ševa. 2019. An extensive review of tools for manual annotation of documents. *Briefings in Bioinformatics*, 22(1):146–163.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria. Association for Computational Linguistics.

Lluís Padró and Evgeny Stanilovsky. 2012. FreeLing 3.0: Towards wider multilinguality. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2473–2479, Istanbul, Turkey. European Language Resources Association (ELRA).

Muhammad Qorib, Seung-Hoon Na, and Hwee Tou Ng. 2022. Frustratingly easy system combination for grammatical error correction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics:*

Human Language Technologies, pages 1964–1974, Seattle, United States. Association for Computational Linguistics.

Muhammad Reza Qorib, Geonsik Moon, and Hwee Tou Ng. 2023. ALLECS: A lightweight language error correction system. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 298–306, Dubrovnik, Croatia. Association for Computational Linguistics.

Alla Rozovskaya and Dan Roth. 2019. Grammar error correction in morphologically rich languages: The case of Russian. *Transactions of the Association for Computational Linguistics*, 7:1–17.

Charanjit Swaran Singh, Amreet Jageer Singh, Nur Qistina Abdul Razak, and Thilaga Ravinthar. 2017. Grammar errors made by ESL tertiary students in writing. *English Language Teaching*, 10:16.

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, Avignon, France. Association for Computational Linguistics.

Regina Stodden and Laura Kallmeyer. 2022. TS-ANNO: An annotation tool to build, annotate and evaluate text simplification corpora. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 145–155, Dublin, Ireland. Association for Computational Linguistics.

Qiongkai Xu, Chenchen Xu, and Lizhen Qu. 2019. ALTER: Auxiliary text rewriting tool for natural language generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 13–18, Hong Kong, China. Association for Computational Linguistics.

Jie Yang, Yue Zhang, Linwei Li, and Xingxuan Li. 2018. YEDDA: A lightweight collaborative text span annotation tool. In *Proceedings of ACL 2018, System Demonstrations*, pages 31–36, Melbourne, Australia. Association for Computational Linguistics.

Helen Yannakoudakis, Øistein E. Andersen, Ardeshir Geranpayeh, Ted Briscoe, and Diane Nicholls. 2018. Developing an automated writing placement system for esl learners. *Applied Measurement in Education*, 31:251–267.

Baoli Zhang, Zhucong Li, Zhen Gan, Yubo Chen, Jing Wan, Kang Liu, Jun Zhao, Shengping Liu, and Yafei Shi. 2021. CroAno : A crowd annotation platform for improving label consistency of Chinese NER dataset. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 275–282, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

A Appendix

A.1 WAMP Modules

WAMP contains the following community-contributed modules:

- Conditional Styles (version 6.x-1.1)
- Hovertip (version 6.x-1.x-dev)
- Jquery Update (version 6.x-2.x).

A.2 Import XML

The import XML file encompasses a list of "instance" elements, as shown in Figure 2. Each "instance" element represents an individual essay to be imported into WAMP and is characterized by a "title" attribute that serves as the title of the specified essay. Under the "instance" element is the "context" element that contains the actual content of the essay. Inside the "context" element, each paragraph is separated by double line breaks and can contain multiple sentences.

A.3 Export XML

The export XML format contains a list of "instance" elements, which represent individual essays imported into WAMP. Each of these "instance" elements, characterized by the "title" attribute, has two main types of child elements, "context" and "annotation." The "context" element comprises the main body of the essay and the "annotation" element contains the annotation XML format. The annotation XML format, shown in Figure 6, is specific to a single essay and by a single annotator. The root element of the annotation XML format is "annotation", which contains a child element "mistake" representing a single annotation. The "start" and "end" attributes denote the start and end position of the specified grammatical error that is annotated. For example, the annotation for "cause," illustrated in Figure 3, is represented as having start attribute of "/0.210" and end attribute of "/0.216." This indicates that the annotation belongs to the 0th line (counting from 0) and the annotation spans from the 210th to 216th character offset positions within that line. Also, the "mistake" element includes three sub-child elements, "type", "correction", and "comments." Each of these sub-child elements contains information about the error type, proposed correction, and comment respectively.

A.4 Docker Dependencies

- Ubuntu (version 9.10)[8]
- Apache (version 2.2)[9]
- PHP (version 5.2)[10]
- MySQL (version 5.1)[11]

[8] https://old-releases.ubuntu.com/releases/karmic/

[9] https://httpd.apache.org/download.cgi

[10] https://www.php.net/releases/index.php

[11] https://downloads.mysql.com/archives/community/