STE Vocabulary Checker and XML-Wrapper for Technical Manuals
Rule-Based Pipeline and Mini Parallel Corpus

Anni Nieminen
Language Technology Resources
University of Gothenburg

Abstract

In this paper, we explore Simplified Technical English (STE), a set of writing guidelines also known as a controlled natural language. Furthermore, we look into Extensible Markup Language (XML), a markup language used especially for effective and organized storing and transmitting of data in documents, readable by both machines and human beings. Having 4 technical manuals extracted from archive.org as our corpora, our main goals are the following:

- creating a script that both flags the words deviating from the set of approved words defined in STE, as well as suggests the user for an approved word
- creating a script that wraps the raw text and the eventual deviations into a nested XML format

Using Python and its appropriate libraries, we create two modules that perform this double task. We evaluate the resulting pipeline's performance by examining the resulting XML-documents and perform error analysis. Our quantitative and qualitative analysis indicate moderate results but offer a base for future work.

1 Introduction and background

1.1 Simplified Technical English (STE)

STE is a type of controlled natural language, or an international standard, which was developed for the composition of various technical documents. Its ultimate goal is to improve safety through L2-friendly use of the English language, given that a majority of professionals dealing with technical manuals or instructions at their work do not speak English as their first language[1]. One of the core principles of STE guidelines is to retain from using more advanced words whenever there is a simpler option available. For instance, the guidelines state that one should not use throughout(prep), but instead use during(prep), as well as cause(v) instead of trigger(v). Additionally, each word can generally only have one POS. Figure 1 below illustrates this.

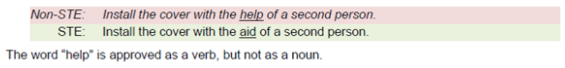

Non-STE: *Install the cover with the help of a second person.*
STE: Install the cover with the aid of a second person.
The word "help" is approved as a verb, but not as a noun.

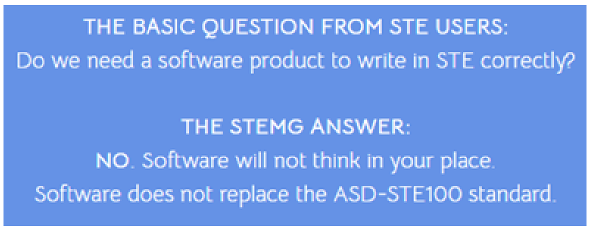
Figure 1: Screen capture of an example of STE rule 9.2. ('Use each word correctly') p. 1-9-6.

Despite STE having been created in the late 70s particularly for the aviation industry, it is now widely adopted in various markets. It is worth noting that STE keeps developing, and its users are welcome to contribute to the redaction of the new versions by sending in requests and suggestions. The guidelines are maintained by the Simplified Technical English Maintenance Group (STEMG), a working group under ASD (Aerospace, Security and Defence Industries Association of Europe).

In addition to the vocabulary restrictions, STE defines a wide range of other rules, having to do with a wide range linguistic aspects such as the correct use of verb forms, writing consistency, sentence length, and punctuation. In this project, however, we are focusing solely on the vocabulary aspect, as we follow the defined set of STE-approved and not-approved words.

As for various tools that might help users in compiling documents in STE, they are usually private or even company-specific STE-supporting software products. Furthermore, as stated on the ASD-STE100 homepage, they are not authorized by ASD. The maintenance group also highlights the importance of using these tools only to enhance the user's existing STE knowledge. Even only using them to flag deviations from STE-guidelines needs to be done with caution.

[1]Simplified Technical English Maintenance Group, www.asd-ste100.org



THE BASIC QUESTION FROM STE USERS:
Do we need a software product to write in STE correctly?

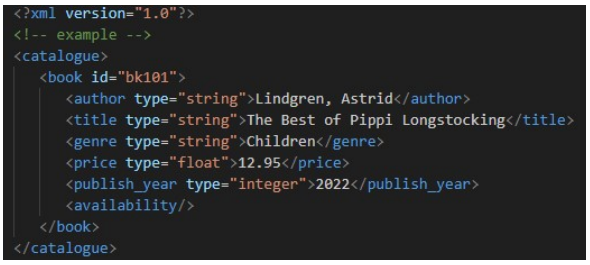
THE STEMG ANSWER:
NO. Software will not think in your place.
Software does not replace the ASD-STE100 standard.

Figure 2: Screen capture from asd-ste100.org/STEsoftware.html

1.2 Extensive Markup Language (XML)

XML is a markup language, more specifically a subset to the Standard Generalized Markup Language, developed in the late 90s. It used to embed data in documents in an organized way, allowing effective storing, retrieving, and transmitting of data. Like other markup languages, XML syntax relies on the use of tags and their appropriate nesting. However, XML differs from, for instance HTML, in that it allows the user to create their own tags[2]. The fundamentals of XML still follow certain standards, creating modularity that the users sharing data benefit from.

The XML guidelines[3] emphasise the two set of criteria for XML documents: Well-Formedness and Validity. The former can be seen as referring to the syntax of the document, for instance whether the document's tags are correctly nested. The latter requires the document to follow a specific external Document Type Definition (DND), which among other things can specify the allowed types for the specific tags in the document. Figure 3 below illustrates an XML document.



```xml
<?xml version="1.0"?>
<!-- example -->
<catalogue>
   <book id="bk101">
      <author type="string">Lindgren, Astrid</author>
      <title type="string">The Best of Pippi Longstocking</title>
      <genre type="string">Children</genre>
      <price type="float">12.95</price>
      <publish_year type="integer">2022</publish_year>
      <availability/>
   </book>
</catalogue>
```

Figure 3: Screen capture of a fictive XML document.

The text inside tags is referred to as elements, although there can also be empty elements, which are defined by an empty-element tag, for instance <availability/> in the figure 3 above. Alongside the name, we can also denote the element's type inside the tag. In figure 3, we have integer, floating point number as well as string types.

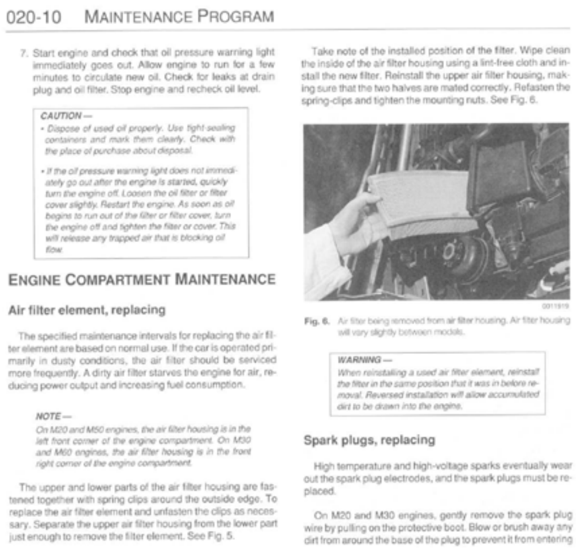
The nested and hierarchical structure of XML-documents allows a structured way of parsing the information. For instance, by using Python's ElementTree- XML support library we can retrieve the root of the document and consequently access information in the children nodes. In the (very simplified) example above the root of the document is the <catalogue> tag, and its children node is the <book> tag etc.

2 Methodology

2.1 Data

The data we used in this project consists of 4 different technical manuals extracted in txt-format from archive.org. These documents are

- Official Honda Shop Manual CB550SC / CB650SC Nighthawk (1984)
- Rainbow NX-1000C Color Printer (1987)
- Bmw 5 Series (E34) Service Manual (1995)
- AN/PEQ-15A DBAL-A2 Dual Beam Aiming Laser Manual (unknown publ. year)



020-10 MAINTENANCE PROGRAM

7. Start engine and check that oil pressure warning light immediately goes out. Allow engine to run for a few minutes to circulate new oil. Check for leaks at drain plug and oil filter. Stop engine and recheck oil level.

CAUTION —
- *Dispose of used oil properly. Use tight-sealing containers and mark them clearly. Check with the place of purchase about disposal.*
- *If the oil pressure warning light does not immediately go out after the engine is started, quickly turn the engine off. Loosen the oil filter or filter cover slightly. Restart the engine. As soon as oil begins to run out of the filter or filter cover, turn the engine off and tighten the filter or cover. This will release any trapped air that is blocking oil flow.*

ENGINE COMPARTMENT MAINTENANCE

Air filter element, replacing

The specified maintenance intervals for replacing the air filter element are based on normal use. If the car is operated primarily in dusty conditions, the air filter should be serviced more frequently. A dirty air filter starves the engine for air, reducing power output and increasing fuel consumption.

NOTE —
On M20 and M50 engines, the air filter housing is in the left front corner of the engine compartment. On M30 and M60 engines, the air filter housing is in the front right corner of the engine compartment.

The upper and lower parts of the air filter housing are fastened together with spring clips around the outside edge. To replace the air filter element and unfasten the clips as necessary. Separate the upper air filter housing from the lower part just enough to remove the filter element. See Fig. 5.

Take note of the installed position of the filter. Wipe clean the inside of the air filter housing using a lint-free cloth and install the new filter. Reinstall the upper air filter housing, making sure that the two halves are mated correctly. Refasten the spring-clips and tighten the mounting nuts. See Fig. 6.

0011919

Fig. 6. Air filter being removed from air filter housing. Air filter housing will vary slightly between models.

WARNING —
When reinstalling a used air filter element, reinstall the filter in the same position that it was in before removal. Reversed installation will allow accumulated dirt to be drawn into the engine.

Spark plugs, replacing

High temperature and high-voltage sparks eventually wear out the spark plug electrodes, and the spark plugs must be replaced.

On M20 and M30 engines, gently remove the spark plug wire by pulling on the protective boot. Blow or brush away any dirt from around the base of the plug to prevent it from entering

Figure 4: Screen capture of one of the technical manuals in our corpora (BMW 5-series service manual).

We manually extracted the maintenance-chapters out of all the manuals. Furthermore, we used Python's pypdf-library to extract text from the STE guidelines (ASD-STE, issue 8). A copy of the

[2] https://www.xml.com/pub/a/1999/01/walsh1.html
[3] https://www.xml.com/axml/axml.html

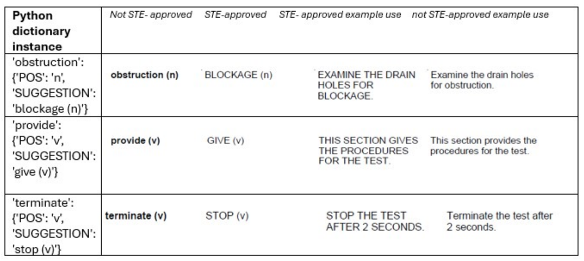
guidelines is requestable from the Simplified Technical English's website by filling a form and sending it as an email to Simplified Technical English Maintenance Group (STEMG).

2.2 Python Scripts

Our scripts are divided into two main modules, which are briefly presented below.

2.2.1 STE Vocabulary Checker

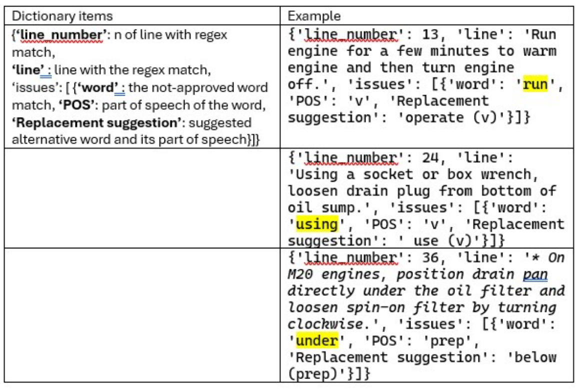
In our first module, we commence by extract data from the ASD-STE100 (issue 8) pdf-file, and creating a data mapping holding not approved words as keys and suggested alternatives as values. This is achieved by regular expression pattern matching. Table 1 below illustrates some of the instances from this dictionary. We conducted a manual verification of suggestions by investigating the STE guidelines.



Python dictionary instance	*Not STE- approved*	*STE-approved*	*STE- approved example use*	*not STE-approved example use*
'obstruction': {'POS': 'n', 'SUGGESTION': 'blockage (n)'}	**obstruction (n)**	BLOCKAGE (n)	EXAMINE THE DRAIN HOLES FOR BLOCKAGE.	Examine the drain holes for obstruction.
'provide': {'POS': 'v', 'SUGGESTION': 'give (v)'}	**provide (v)**	GIVE (v)	THIS SECTION GIVES THE PROCEDURES FOR THE TEST.	This section provides the procedures for the test.
'terminate': {'POS': 'v', 'SUGGESTION': 'stop (v)'}	**terminate (v)**	STOP (v)	STOP THE TEST AFTER 2 SECONDS.	Terminate the test after 2 seconds.

Figure 5: Examples from the created dictionary.

Then, we read the input files, match lines of the input documents against a regular expression that contains the not-approved words. Subsequently, we create a dictionary holding information about the found not-approved words in the document and the suggested replacements extracted from the STE document. For POS-tagging the words in the input files we use spaCy's "en core web sm" language pipeline. This creates more consistency in the flagging of the deviations (the POS of the flagged word and suggestion need to match). Figure 6 below shows examples of this stage.



Dictionary items	Example
{**'line_number'**: n of line with regex match, **'line'** : line with the regex match, 'issues': [{**'word'** : the not-approved word match, **'POS'**: part of speech of the word, **'Replacement suggestion'**: suggested alternative word and its part of speech}]}	`{'line_number': 13, 'line': 'Run engine for a few minutes to warm engine and then turn engine off.', 'issues': [{'word': 'run', 'POS': 'v', 'Replacement suggestion': 'operate (v)'}]}`
	`{'line_number': 24, 'line': 'Using a socket or box wrench, loosen drain plug from bottom of oil sump.', 'issues': [{'word': 'using', 'POS': 'v', 'Replacement suggestion': ' use (v)'}]}`
	`{'line_number': 36, 'line': '* On M20 engines, position drain pan directly under the oil filter and loosen spin-on filter by turning clockwise.', 'issues': [{'word': 'under', 'POS': 'prep', 'Replacement suggestion': 'below (prep)'}]}`

Figure 6: STE-deviating lines of the input document.

2.2.2 XML-Wrapper

In this module, we import the flagged sentences from STE-checker module. In order to format our data in XML, we work with ElementTree XML API Python library, which can be used to both parse and build XML documents.

Subsequently, we create a nested structure where paragraphs are organized according to their headings (usually capitalized in the raw txt.-documents), the body of the text is categorized to either "instruction" or "warning", based on the use of certain key words. Furthermore, using key-word matching, we extract the tools/objects from the body into another nested tag structure. This allows the user to extract the tools-node from the manual. However, this feature was added in the later stages of our project and is mainly useful for only the Honda and BMW manuals. We would also like to note that due to our own insufficient knowledge of the variety of different tools, we used a GPT-model in order to enhance our original list of tools against which the input files are compared against (this file is available in our GitHub repository, which can be found under appendices).

It is important to note that the original pdf-format manuals in our corpora contain many images that accompany the instructions. These images are not available in the txt-files.

3 Results and Evaluation

3.1 Evaluation of the Parallel Corpus

We will start with a brief quantitative analysis of the STE-checker module, analyzing the accuracy of the word replacement suggestions. For this, we chose the 50 first sentences of each of our corpus. After that, we will perform a small error analysis of some examples. Subsequently, we will move on to briefly visually inspect the performance of the XML-wrapper module.

By accuracy in terms of the word replacements we refer to such suggestions that a) have matching POS-tags and b) form a (grammatically and semantically) valid sentence. From table 1 below we can see that we reach an average of 59% suitable word replacement suggestions. We will introduce examples of correctly flagged sentences in chapter 4,

Corpus and % of correct word replacements
Honda Manual 55%
Color Printer 63%
Dual Beam Laser 66%
BMW Series 5 53%

Table 1: Performance of the STE-vocabulary checker.

figure 9.

3.1.1 Manual Qualitative Inspection of Errors

In this subsection, we will provide examples of two of the most prominent error types that we found whilst examining the performance of the STE-checker module.

3.1.2 Flagged word has a wrong POS-tag

The examples below illustrate how the SpaCy library we used for POS-tagging the input sentences doesn't manage to correctly tag all the input words. This leads to incorrect word suggestions.

Example 1. *('Plug the vacuum tube end with a suitable stopper. Start the engine and adjust the idle speed. IDLE SPEED: 1,100 + 100 rpm', 'word': 'end', 'POS': 'v', 'Replacement suggestion': 'stop (v)')*

Example 2. *('Turn the throttle stop screw as required to obtain the specified idle speed.', 'word': 'screw', 'POS': 'v', 'Replacement suggestion': 'turn (v)')*

Example 3. *('+ Use gloves to protect your hands.', 'word': 'use', 'POS': 'n', 'Replacement suggestion': 'operation (n)')*

3.1.3 Word meaning out of context or unrecognized

Example 4. *('6. Refill crankcase with oil. Check oil level when finished adding oil. Oil specifications are found under Fluid and Lubricant Specifications.', 'word': 'under', 'POS': 'prep', 'Replacement suggestion': 'below (prep)')*

In the example 4 above, the word replacement suggestion is incorrect. The STE guidelines document lacks an example of this type of use of the word under.

Example 5. *('NOTE —'The upper and lower parts of the air filter housing are fas- tened together with spring clips around the outside edge ', 'word': 'note', 'POS': 'v', 'Replacement suggestion': 'record (v)')*

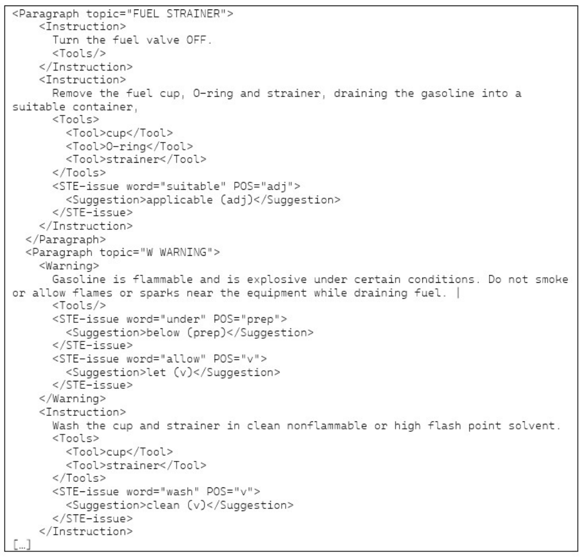
The STE guidelines have "note (v.)" marked as not approved word and suggest using record (v.) instead. However, in example 5 above, this replacement is out of context.

Example 6. *('Use a spark plug gap gauge to check the gap, if applicable. If necessary, bend the outer electrode slightly to adjust the gap to meet the specifica-tion.' 'word': 'meet', 'POS': 'v', 'Replacement suggestion': 'engage (v)')*

The STE guidelines only use the term "meet (v.)" in the sense of touching. Consequently, in example 6, the suggestion is not valid.

3.1.4 Manual Qualitative Evaluation of the XML-wrapper

We will start by introducing an example snippet of one the XML-document created by our XML-wrapper module. We will follow that with a brief discussion and error analysis.



```
<Paragraph topic="FUEL STRAINER">
    <Instruction>
      Turn the fuel valve OFF.
      <Tools/>
    </Instruction>
    <Instruction>
      Remove the fuel cup, O-ring and strainer, draining the gasoline into a
suitable container,
      <Tools>
        <Tool>cup</Tool>
        <Tool>O-ring</Tool>
        <Tool>strainer</Tool>
      </Tools>
      <STE-issue word="suitable" POS="adj">
        <Suggestion>applicable (adj)</Suggestion>
      </STE-issue>
    </Instruction>
  </Paragraph>
  <Paragraph topic="W WARNING">
    <Warning>
      Gasoline is flammable and is explosive under certain conditions. Do not smoke
or allow flames or sparks near the equipment while draining fuel. |
      <Tools/>
      <STE-issue word="under" POS="prep">
        <Suggestion>below (prep)</Suggestion>
      </STE-issue>
      <STE-issue word="allow" POS="v">
        <Suggestion>let (v)</Suggestion>
      </STE-issue>
    </Warning>
    <Instruction>
      Wash the cup and strainer in clean nonflammable or high flash point solvent.
      <Tools>
        <Tool>cup</Tool>
        <Tool>strainer</Tool>
      </Tools>
      <STE-issue word="wash" POS="v">
        <Suggestion>clean (v)</Suggestion>
      </STE-issue>
    </Instruction>
[…]
```

Figure 7: Example snippet from Honda manual transformed into XML with our wrapper.

Figure 7 above illustrates the structure of the XML-file created from the Honda manual. "FUEL STRAINER" and "W WARNING" are categorized as the topics of the paragraphs, and following lines are classified to either instruction or warning. Some tools are gathered under the <tools> node, with the goal of facilitating the eventual retrieval of necessary tools in each step. Furthermore, the vocabulary that deviates from STE-approved words are nested inside the “STE-issue” tags. Some of the most notable errors we encountered after having conducted a visual qualitative analysis on the XML-documents were the following:

- The line breaks and missing images/figures in the input txt-files caused the structure of the XML-files to be occasionally unorganized. For example, instruction sentences that could have been grouped into one paragraph (under the same <instruction> or <warning> tag) were separated, causing the node to get excessively long. This could be solved by additional work into analysing the input files.

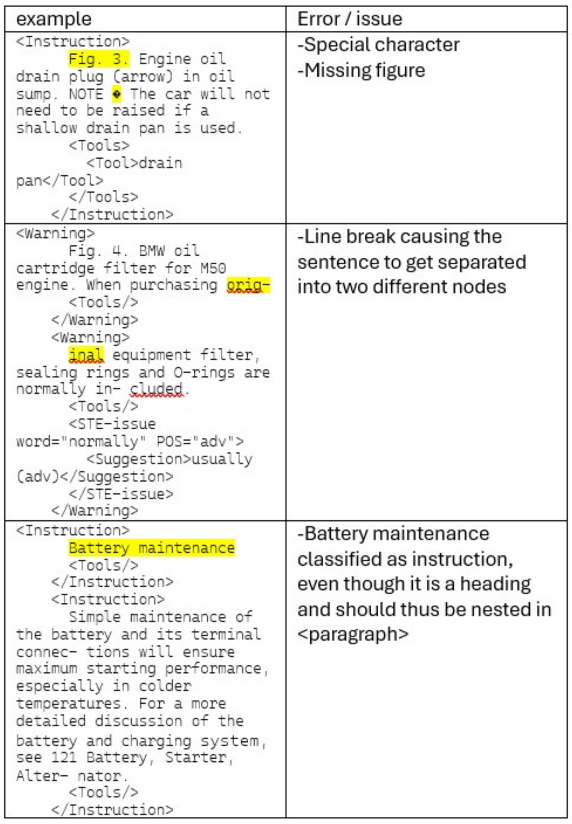
- The missing images and figures caused the input files to contain some special characters that were not utf-8 compatible.

- Hard coding some keywords and relying on binary checking of capitalization in classifying lines of text nor items is not the safest option. For instance, in the XML-document created from the BMW-manual, some of the text lines that were intended as the headings were misclassified as the body.

As we stated in chapter 2, XML-documents are usually evaluated after their Well-Formedness and Validity. In our project, we have only been interested in the Well-Formedness, as none of the resulting XML-documents do not contain Document Type Definitions, which are required for the XML-document to be considered valid. We can state that our XML-documents were indeed well-formed, but we also need to keep in mind that in this project, we were dealing with a relatively low number of different tags.

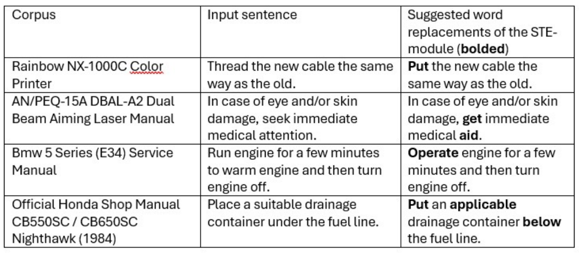
4 Discussion

As we saw from the results, in terms of accuracy of the word replacement suggestions of STE-checker module, its performance leaves plenty of room for improvement. However, we are still pleasantly surprised that a simple script like ours, relying solely on a rule-based approach can reach an accuracy of over 50%. As we noted in the manual qualitative inspection of errors, one of the most prominent challenge rises from the POS-tagging of the input sentences. The structural format of the corpora also presents its issues. This is due to the txt-documents having been created from pdf-files,



example	Error / issue
`<Instruction>` `Fig. 3. Engine oil drain plug (arrow) in oil sump. NOTE ◆ The car will not need to be raised if a shallow drain pan is used.` `<Tools>` `<Tool>drain pan</Tool>` `</Tools>` `</Instruction>`	-Special character -Missing figure
`<Warning>` `Fig. 4. BMW oil cartridge filter for M50 engine. When purchasing orig-` `<Tools/>` `</Warning>` `<Warning>` `inal equipment filter, sealing rings and O-rings are normally in- cluded.` `<Tools/>` `<STE-issue word="normally" POS="adv">` `<Suggestion>usually (adv)</Suggestion>` `</STE-issue>` `</Warning>`	-Line break causing the sentence to get separated into two different nodes
`<Instruction>` `Battery maintenance` `<Tools/>` `</Instruction>` `<Instruction>` `Simple maintenance of the battery and its terminal connec- tions will ensure maximum starting performance, especially in colder temperatures. For a more detailed discussion of the battery and charging system, see 121 Battery, Starter, Alter- nator.` `<Tools/>` `</Instruction>`	-Battery maintenance classified as instruction, even though it is a heading and should thus be nested in <paragraph>

Figure 8: Some examples of errors in the XML-file of the BMW manual.



Corpus	Input sentence	Suggested word replacements of the STE-module (**bolded**)
Rainbow NX-1000C Color Printer	Thread the new cable the same way as the old.	**Put** the new cable the same way as the old.
AN/PEQ-15A DBAL-A2 Dual Beam Aiming Laser Manual	In case of eye and/or skin damage, seek immediate medical attention.	In case of eye and/or skin damage, **get** immediate medical **aid.**
Bmw 5 Series (E34) Service Manual	Run engine for a few minutes to warm engine and then turn engine off.	**Operate** engine for a few minutes and then turn engine off.
Official Honda Shop Manual CB550SC / CB650SC Nighthawk (1984)	Place a suitable drainage container under the fuel line.	**Put** an **applicable** drainage container **below** the fuel line.

Figure 9: Examples of correct word replacements.

leading to the organization of the document not being the clearest.

When it comes to the XML-wrapper and the output files, we can be at least moderately satisfied with its performance. The XML-files are nested in a correct way, but additional work on the input files is required to further refine the organization and logic.

5 Conclusion

In this paper, we have attempted to create a small and very tentative pipeline that could be used in attempts to automatize a subpart of translating manuals into STE. We believe that in spite of STE being mainly used for documents that are not directed to consumers, consumer manuals could also benefit from a clear and simple English. This project was merely a preliminary attempt at familiarizing ourselves with these guidelines and frameworks.

After having carefully analyzed a sample of the flagged deviations, we were able to see that on average approximately 59% of the word replacement suggestions were valid.

Qualitative analysis raised multiple challenges, such as the flagged words being tagged with a wrong part of speech, leading to incorrect word replacement suggestions. It goes also without saying that the XML-wrapper module requires further work in order to better grasp the structure of the manuals, and to rely less on rigid pattern-matching.

Our approach has been heavily rule-based and rather unflexible. In the future, it might be interesting to possibly explore utilizing an LLM in the process of flagging the word deviations. By carefully refining the prompts, the LLM might be able to do a better job at dealing with the POS-tagging of the input sentences, leading to improved results in suggesting replacement words. However, as we are dealing with technical documents and the correct use of words is of utmost importance, we believe that a somewhat hybrid system combining rule-based approaches might be the most suitable option.

Furthermore, as we stated earlier, despite there not being many available tools for transforming documents into STE-format, it would be interesting to explore these tools and the approaches they have adopted.

6 References

Simplified Technical English Maintenance Group Website, *ASD-STE100 Homepage*, 2025. Available: https://www.asd-ste100.org. Accessed: 13th of January, 2025.

Mozilla Developer Network, *XML Introduction*, 2025. Available: https://developer.mozilla.org/en-US/docs/Web/XML/XML_introduction. Accessed: 13th of January, 2025.

XML.com, *A Guide to XML*, 1998. Available: https://www.xml.com/axml/axml.html. Accessed: 13th of January, 2025.

ASD-STE100, Simplified Technical English, *ASD-STE, Issue 8*, 2021. Requestable from ASD-STE100 Homepage.

Python Software Foundation Documentation, *xml.etree.ElementTree – The ElementTree XML API*, 2025. Available: https://docs.python.org/3/library/xml.etree.elementtree.html. Accessed: 13th of January, 2025.

6.1 Appendices

The Python scripts, corpora, as well as other files of this project can be found at the following open repository: https://github.com/Anurni/LanguageResources