TEASPN: Framework and Protocol for Integrated Writing Assistance Environments

Masato Hagiwara[1] **Takumi Ito**[2,3] **Tatsuki Kuribayashi**[2,3]
Jun Suzuki[2,4] **Kentaro Inui**[2,4]
[1]Octanove Labs LLC [2]Tohoku University [3]Langsmith Inc. [4]RIKEN AIP
masato@octanove.com
{t-ito, kuribayashi, jun.suzuki, inui}@ecei.tohoku.ac.jp

Abstract

Language technologies play a key role in assisting people with their writing. Although there has been steady progress in e.g., grammatical error correction (GEC), human writers are yet to benefit from this progress due to the high development cost of integrating with writing software. We propose TEASPN[1], a protocol and an open-source framework for achieving integrated writing assistance environments. The protocol standardizes the way writing software communicates with servers that implement such technologies, allowing developers and researchers to integrate the latest developments in natural language processing (NLP) with low cost. As a result, users can enjoy the integrated experience in their favorite writing software. The results from experiments with human participants show that users use a wide range of technologies and rate their writing experience favorably, allowing them to write more fluent text.

1 Introduction

Language technologies have been playing an important role in assisting people in writing natural language texts, such as essays, emails, business documents, and academic papers. There has been considerable progress on writing assistance technologies (or *WATs* in short) in the past few decades in fields such as NLP and computer-aided language learning (CALL). For example, in one of such areas, grammatical error correction (GEC) (Leacock et al., 2010), new models and systems are developed and published month after month, breaking the previous evaluation records and advancing state of the art. The recent development in neural language models enabled the completion of a prompt with long, realistic looking yet coherent passages (Radford et al., 2019).

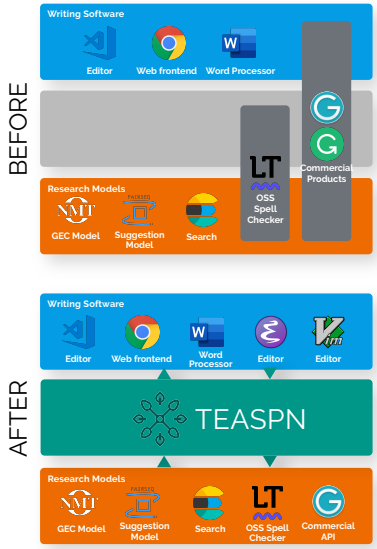

Figure 1: Writing software before and after TEASPN.

However, real-world users such as writers who can and should benefit the most from WATs are yet to reap the fruits from these research efforts. Aside from a small number of commercial products, notably Grammarly[2] and Smart Compose (Chen et al., 2019), and research systems such as WriteAhead (Yen et al., 2015; Chang and Chang, 2015) and CroVeWA (Soyer et al., 2015), we see few examples of user-facing applications and experiments that make use of recent development in WATs. Many models are confined in research implementations that are not easily accessible to end users and the larger society. WATs, however, are not truly useful until they are integrated into user-facing writing applications such as editors and word processors (collectively called *writing software* in this paper) and interact with end users in a dynamic and intuitive manner. This "great divide" (see Figure 1 BEFORE) between applica-

[1]See https://www.teaspn.org/demo for the screencast and https://www.teaspn.org/ for more general info about TEASPN.

[2]https://www.grammarly.com/

Proceedings of the 2019 EMNLP and the 9th IJCNLP (System Demonstrations), pages 229–234
Hong Kong, China, November 3 – 7, 2019.

tions and academia is not unique in the domain of writing assistance, but a widespread phenomenon across many fields in machine learning and NLP, as pointed out by Wagstaff (2012).

One cause of this "great divide" is the high development cost for integrating and bridging both sides. Since there is a wide range of WATs, it is impractical, if not impossible, for developers of writing software to support all types of such technologies that come in different packages in different programming languages. Similarly, since there is a large selection of writing software, WAT researchers and developers cannot afford to offer their solutions in such a way that most writing software packages can benefit from them. If there are N types of writing software and M types of WATs, there can be $N \times M$ combinations between the two sides. As a result, writers often need to rely on many different writing software solutions and switch between many different applications and websites (search engines, grammar checkers, dictionaries and thesauri, etc.) in order to complete their tasks.

In this paper, we propose TEASPN (Text Editing Assistance Smartness Protocol for Natural Language; pronounced "teaspoon"), a protocol and a framework for achieving integrated writing assistance environments, as a solution to this "great divide" problem (Figure 1 AFTER). Inspired by and built upon Language Server Protocol (LSP)[3], a similar protocol for integrating software development environments, TEASPN provides an open protocol that standardizes the way writing software and WATs communicate with each other. We also released the TEASPN SDK (software development kit) as an open source library, which eases the cost of making WATs compatible with TEASPN. As a result, by using TEASPN,

- Developers of writing software can easily integrate state-of-the-art WATs into their editors and word processors just by following the protocol.
- Developers and researchers of WATs can support major writing software applications without worrying about the development cost, just by using the TEASPN SDK.
- Writers can benefit from integrated writing experience provided by their favorite writing software and WATs.

[3] https://microsoft.github.io/language-server-protocol/

Finally, we implemented a demo TEASPN server that integrates WATs using latest developments in NLP (e.g., a neural language model and seq2seq-based paraphrasing) and ran experiments with real human writers to verify the framework's effectiveness. The experimental results demonstrated that our integrated writing assistance system developed with TEASPN provides better writing experience for human writers.

2 Related Work

Writing assistance Use of language technologies for assisting writing in a second language (L2) has been extensively explored, especially for non-native English speakers. One of the most active research areas is GEC (Leacock et al., 2010), where several new models are published every year and commercial systems such as Grammarly are actively developed. Other research-based systems include WriteAhead (Yen et al., 2015; Chang and Chang, 2015), an interactive writing environment that provides users with grammatical patterns mined from large corpora, and CroVeWA (Soyer et al., 2015), a crosslingual sentence search system for L2 writers. FLOW (Chen et al., 2012) is another writing assistance system that allows users to type in their first languages (L1) and suggests words and phrases in L2. Running syntactic analysis and visualizing sentence structures have also been explored for L2 reading assistance (Faltin, 2003; Srdanović, 2011).

In addition to L2 learners, the use of technologies for assisting human translators has also been a focus of research. TransType (Langlais et al., 2000) is a translation assistance system that suggests completions for the text to the human translator in an interactive manner. In SemEval 2014, van Gompel et al. (2014) presented an L2 writing assistance task where systems find the proper translation of a word given a context in L2. Other writing assistance systems (not necessarily L2 learners) include assisting users with composing an email by auto-completion (Chen et al., 2019) and reply suggestion (Kannan et al., 2016).

LSP The $N \times M$ problem mentioned in Section 1 is not unique to writing assistance. In software development, there can be N different types of integrated development environments (IDEs) and M different programming languages, making the integration cost proportionally expensive to $N \times M$. LSP solved this problem by

proposing an open protocol that standardizes the way IDEs communicate with servers that offer language smartness technologies such as syntax checking and completion. As of today, LSP is widely adopted and supported by more than 70 servers and 20 development environments.

Since there is a large overlap between authoring in programming and natural languages, we built TEASPN as a "fork" of LSP. There are a few features that we need to design and implement specifically for writing assistance, namely, syntax highlighting and external resource search, which makes TEASPN incompatible with LSP. However, we re-purposed many LSP data models and features for TEASPN. Being able to leverage existing resources for LSP gives TEASPN a great head start for a wide adoption.

Protocols for NLP Language Grid (Ishida, 2006) is a platform where language providers (e.g., translation systems) and linguistic resources (e.g., dictionaries) are connected via semantic Web technologies to provide language services to communities. The NLP Interchange Format (NIF) (Hellmann et al., 2012) is a standard that aims to achieve interoperability between different NLP tools and resources by defining an RDL/OWL-based format. Although these projects have seen some real-world success, their adoption is quite limited as of this writing, compared to the aforementioned LSP, which powers at least a couple of millions of developers worldwide both for Visual Studio Code (VS code)[4] and Atom[5]. We believe the key to the wide adoption of any protocol is the focus on the right scope, practicality, and ease of development, which are the guiding principles for TEASPN.

3 TEASPN

3.1 Overview

TEASPN adopts a client-server architecture (Figure 2), where a client (writing software such as an editor or a word processor) communicates with a server that provides WATs. The client and the server communicate over the TEASPN protocol, an HTTP-like protocol which uses JSON-RPC (remote procedure call)[6] to encode the message body.

[4] https://code.visualstudio.com/blogs/2017/11/16/connect

[5] https://blog.atom.io/2016/03/28/atom-reaches-1m-users.html

[6] http://www.jsonrpc.org/

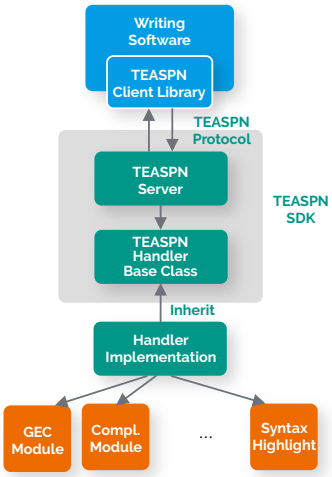

Figure 2: Architecture of TEASPN.

Requests can be sent in both directions, which are often triggered by some events (such as user input) and can be responded with additional data (such as results of GEC). TEASPN clients and servers can be written in any programming language as long as they conform to the protocol.

3.2 Features

By integrating a large selection of WATs in a single platform, TEASPN makes them available to writers at their fingertips and is expected to improve writing effectiveness. Table 1 shows the list of WATs that are supported by TEASPN. Notice that the list includes a wide range of WATs that have been extensively investigated the literature (e.g., GEC and search) as well as the ones that are less explored (e.g., syntax highlighting and jump).

Although we were able to build many WAT features upon existing ones from LSP, there were two features that we needed to design from scratch specifically for TEASPN—syntax highlighting and search. While syntax highlighting is usually handled on the client side for programming languages using shallow lexical analysis, syntactic analyses of natural language can be too costly and complex to be handled solely by the client. Therefore, we defined a new type of request and related type declarations so that it can be handled by the server.

The second feature, search, enables writers to search external linguistic resources. While the search feature for LSP is limited to the files in the same workspace, writers of natural language texts often need to consult a wide variety of resources such as corpora and dictionaries.

Feature	Description
Syntax highlighting	Highlighting parts of text
Grammatical error detection (GED)	Detecting typological and grammatical errors
Grammatical error correction (GEC)	Automatically correcting issues detected by GED
Completion	Completing or suggest succeeding text
Text rewriting	Rewriting part of text (paraphrasing, translation, etc.)
Jump	Jumping to other locations (coreference, definitions, etc.)
Hover	Showing extra information about the location (e.g., definition)
Search	Searching external resources such as corpora and dictionaries

Table 1: WATs supported by TEASPN.

3.3 Developers of Writing Software

By adopting TEASPN, developers of writing software can easily integrate WATs into their editors and word processors. The fact that a large number of IDEs and editors already support LSP helps to a great extent. For example, in the sample TEASPN client implementation[7] we provide for VS Code, we needed to modify less than 200 lines of TypeScript code to make it compatible with the TEASPN protocol while leveraging the existing library for LSP. We were also able to implement preliminary TEASPN clients for Atom and Sublime text[8] with little modification to existing code.

3.4 Developers of WATs

Developers and researchers of WATs can make their technologies available to major writing software just by using TEASPN without writing any client code. To facilitate the development process, we released the TEASPN SDK, which includes a library and a sample TEASPN server implementation in Python, one of the most popular programming languages for developing language technologies as of late. The library takes care of low-level communication and text synchronization with the client, letting WAT developers just inherit the TEASPN handler base class and focus on implementing the missing core NLP logic. As an example, Figure 3 shows a simplified code snippet for implementing completion. Notice the brevity of the code. We also provide a simple yet working TEASPN server implementation in the SDK for reference to accelerate the development.

```
@overrides
def get_completion_list(
      self, position: Position) -> CompletionList:
    """Handle autocomplete."""
    offset = self._position_to_offset(position)
    context = self._text[:offset]

    items = []
    # ... add completion items ...
    return CompletionList(isIncomplete=True, items=items)
```

Figure 3: Code snippet for computing completion

4 Experiments and Implementation

4.1 Experiments

In this section, we develop a demo TEASPN system and investigate its effectiveness through experiments with end-users (writers) in order to answer the following research question:

Does the integrated writing assistance environment developed with TEASPN provide better writing experience and help write better texts?

To explore the effectiveness of integrated writing assistance environment, we compared the following two conditions. In the **INTEGRATED** condition, participants used an editor (VS Code) equipped with TEASPN, where many WATs were available, while in the **BASELINE** condition, they used the same editor with no WATs activated, while being allowed to use any other writing tools outside the editor (e.g., Grammarly and Web dictionaries). The BASELINE condition was set up to simulate the real situation that writers face, where writing assisting technologies are implemented separately outside the editor.

The participants of our experiments consist of twelve college students or researchers in NLP with a diverse L1 distribution: Bengali: 1, Chinese: 1, Croatian: 1, German: 1, Hindi: 1, Japanese: 6, Spanish: 1. They were directed to go through two writing sessions, one for each condition mentioned above, during which they wrote English text within five sentences in response to two different prompts: (i) *write about an activity you enjoy, such as a hobby*, and (ii) *write about your hometown*. The prompts and the writing environments were combined randomly. Before the writing sessions came an instruction session, where

[7] https://github.com/teaspn/teaspn-sdk
[8] https://www.sublimetext.com/

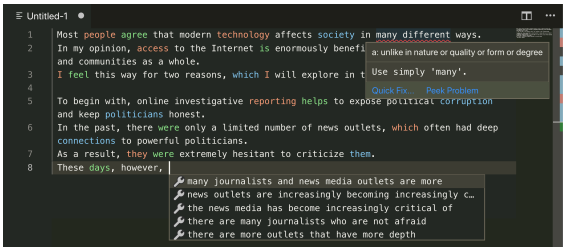

Figure 4: Screenshot of our demo system.

the authors of this paper showed the participants all the features of the demo system for a demonstration purpose. After the writing sessions, they were asked questions regarding their writing experience. Because the focus of this experiment is to evaluate the integrated writing assistance environment, participants are instructed not to consider the performance of individual WATs.

4.2 Implementation of the Demo System

We implemented a demo system using the TEASPN framework which has all of the features shown in Table 1. See Figure 4 for the screenshot.

For syntax highlighting, we used the dependency parser SpaCy[9]. Head tokens with specific dependency relation[10] were highlighted in different colors. As for the GEC and GED features, we used the open-source GEC tool LanguageTool 3.2[11]. We implemented two types of completion features: one which suggests the likely next phrases given the context using a neural language model (Radford et al., 2019) and the other one which suggests a set of words consistent with the characters being typed. We built a seq2seq paraphrase model trained on PARANMT-50M (Wieting and Gimpel, 2018) for the text rewriting feature, which allows the writer to select a part of the text and chooses among paraphrases. As for the jump feature, we used a coreference resolution model[12] to jump from a selected expression to its antecedent. The hover feature shows the definition of a hovered word using WordNet[13]. Finally, we implemented a full-text search feature using the open multilingual sentence dataset Tatoeba[14] and used Elasticsearch 7.1.1[15] for indexing and search.

[9] https://spacy.io/

[10] ROOT, nsubj, nsubjpass, and dobj in the CLEAR style tag set.

[11] https://github.com/languagetool-org/languagetool/releases/tag/v3.2

[12] https://github.com/huggingface/neuralcoref

[13] https://wordnet.princeton.edu/

[14] https://tatoeba.org/eng/

[15] https://www.elastic.co

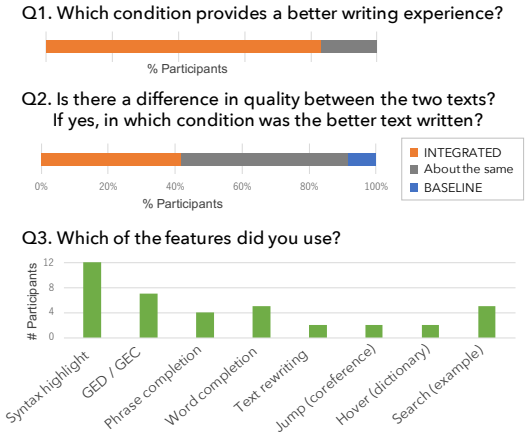

Figure 5: Response summary of the questionnaire.

Condition	Perplexity	# Chars. (mean ± std)
BASELINE	37.8	379 ± 116
INTEGRATED	26.4	335 ± 91

Table 2: Statistics of the written texts.

5 Results and Analysis

After the writing sessions, the participants responded to a questionnaire including the following questions: *Q1: Which environment provides a better writing experience?*, *Q2: Is there a difference in quality between the two texts? If yes, in which environment was the better text written?*, and *Q3: Which of the following features did you use?*

Figure 5 summarizes the responses from the participants. Ten out of twelve (83.3%) participants rated their experience favorably (Q1), and 40% believed they were able to write better texts in the INTEGRATED condition (Q2), demonstrating the effectiveness of the integrated writing assistance environment with TEASPN. The responses for Q3 show that an average participant used 3.2 WAT features in the INTEGRATED condition[16]. This suggests that the writers can benefit from an integrated environment with various WATs activated.

We ran further analyses on the texts written by the participants during the writing sessions. Table 2 shows some statistics of the written texts in the two experimental conditions. Perplexity was calculated using the pretrained GPT-2 model (small, 117M parameters) (Radford et al., 2019). The texts written in the INTEGRATED condition had lower perplexity, suggesting that the integrated writing environment helped them write

[16] Note that we assume that every participant used syntax highlighting, which is activated by default.

more fluent and/or typical English text. This result backs up the subjective responses from the participants indicating they were able to produce better texts in the INTEGRATED condition than the other. We also note that the texts written in the INTEGRATED condition were relatively shorter. This could be due to the fact that the participants were still spending some of their time observing and figuring out the behavior of the assisting features and spending slightly less time actually writing. We believe this trend will disappear or even reverse itself as they get more used to the integrated writing experience and the quality of the inidividual WATs improve.

6 Conclusion and Future Work

We proposed TEASPN, a framework and a protocol which standardizes the way writing software communicates with writing assistance technologies, to achieve integrated writing assistance environments. In addition, we released the TEASPN SDK as an open source library, which eases the cost of making WATs compatible with TEASPN. We developed a demo system which implements various assistance technologies based on latest NLP developments and ran experiments with human participants. The result demonstrated that they rated their integrated writing experience favorably, potentially helping them write more fluent and better text.

In future work, by making this a larger community effort, we wish to broaden the support lineup for writing software while developing various writing assistance features with TEASPN, further closing the gap between the latest developments in NLP and real-world human users.

References

Jim Chang and Jason Chang. 2015. WriteAhead2: Mining lexical grammar patterns for assisted writing. In *Proceedings of the NAACL 2015 System Demonstrations*, pages 106–110.

Mei-Hua Chen, Shih-Ting Huang, Hung-Ting Hsieh, Ting-Hui Kao, and Jason S. Chang. 2012. FLOW: A first-language-oriented writing assistant system. In *Proceedings of ACL 2012 System Demonstrations*, pages 157–162.

Mia Xu Chen, Benjamin N Lee, Gagan Bansal, Yuan Cao, Shuyuan Zhang, Justin Lu, Jackie Tsay, Yinan Wang, Andrew M. Dai, Zhifeng Chen, Timothy Sohn, and Yonghui Wu. 2019. Gmail smart compose: Real-time assisted writing.

Anne Vandeventer Faltin. 2003. Natural language processing tools for computer assisted language learning. *Linguistik Online*, 17(5):137–153.

Maarten van Gompel, Iris Hendrickx, Antal van den Bosch, Els Lefever, and Véronique Hoste. 2014. SemEval 2014 task 5 - L2 writing assistant. In *Proceedings of SemEval 2014*, pages 36–44.

Sebastian Hellmann, Jens Lehmann, and Soren Auer. 2012. NIF: An ontology-based and linked-data-aware NLP interchange format. Technical report, Working Draft.

Toru Ishida. 2006. Language Grid: An infrastructure for intercultural collaboration. In *IEEE/IPSJ Symposium on Applications and the Internet (SAINT 2006)*, pages 96–100.

Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Gregory S. Corrado, László Lukács, Marina Ganea, Peter Young, and Vivek Ramavajjala. 2016. Smart reply: Automated response suggestion for email. In *Proceedings of KDD 2016*.

Philippe Langlais, George Foster, and Guy Lapalme. 2000. TransType: a computer-aided translation typing system. In *ANLP-NAACL 2000 Workshop: Embedded Machine Translation Systems*.

Claudia Leacock, Martin Chodorow, Michael Gamon, and Joel Tetreault. 2010. *Automated Grammatical Error Detection for Language Learners*. Morgan & Claypool.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Technical report, OpenAI.

Hubert Soyer, Goran Topić, Pontus Stenetorp, and Akiko Aizawa. 2015. CroVeWA: Crosslingual vector-based writing assistance. In *Proceedings of NAACL 2015 System Demonstrations*, pages 91–95, Denver, Colorado.

Irena Srdanović. 2011. Evaluating e-resources for Japanese language learning. In *Proceedings of eLex 2011*, pages 260–267.

Kiri Wagstaff. 2012. Machine learning that matters. In *Proceedings of the Twenty-Ninth International Conference on Machine Learning (ICML)*, pages 529–536.

John Wieting and Kevin Gimpel. 2018. ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of ACL 2018*, pages 451–462.

Tzu-Hsi Yen, Jian-Cheng Wu, Jim Chang, Joanne Boisson, and Jason Chang. 2015. WriteAhead: Mining grammar patterns in corpora for assisted writing. In *Proceedings of ACL-IJCNLP 2015 System Demonstrations*, pages 139–144.