Recent Advancements of the New Online Proofreader of Czech

Vojtěch Mrkývka

Faculty of Arts, Masaryk University
Arne Nováka 1, 602 00 Brno, Czech Republic
mrkyvka@phil.muni.cz

Abstract. On the previous RASLAN workshop, the basis of the new online proofreader for the Czech language was presented. This paper describes the current status quo of this tool as well as describes changes necessary due to alterations of the assignment.

Keywords: grammar checker, text analysis, proofreading, Czech

1 Introduction

The new online proofreader is being developed at the Masaryk University (with the help from external experts) since 2018. The motivation to do so is the parallel development of separate rulesets for finding different types of mistakes in the Czech language[3] (for example spelling, commas or agreement) using SET analyser[1] as well as lack of real proofreading tool for the online environment. The previous paper, *Towards the New Czech Grammar-checker*[1][2], compared the new proofreader with the proofreading capabilities of Microsoft Office Word as described in *Kontrola české gramatiky (český grammar checker)* by Vladimír Petkevič[4], seeing differences in approach to mistakes and their highlighting as well as to the licencing of the final product.

2 The original approach

The first version of the proofreader was implemented as part of the TinyMCE online text processor.[2] It consisted of separate modules with limited mutual dependency (see fig. 1), but with strong connections to the editor itself.[3] The major part was written in JavaScript with connection to Python backend when necessary.

[1] There was a dispute whether to use expression grammar-checker, Grammar Checker, proofreader or something else as authors of different components used this inconsistently. In May of 2019, the consensus was reached and the Online Proofreader is used since. Sorry for the inconvenience.

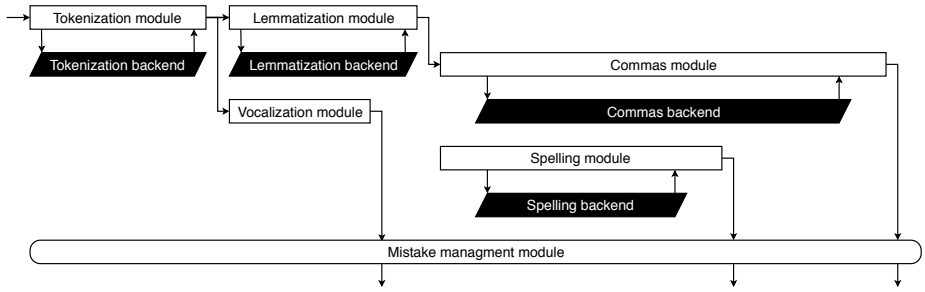
[2] `https://www.tiny.cloud/`

[3] For a more detailed description read already mentioned *Towards the New Czech Grammar-checker*.

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2019, pp. 43–47, 2019.

The main advantage of this approach was that the different components (existing and new) could have been independently developed and also the new versions of these components could be independently released. This, however, led to problems when the need of improvement touched some of the core functionalities (such as the mistake manager). Additionally, this approach was unpleasant from the end-user point of view as the installation and maintaining process was not very intuitive.

Fig. 1: The workflow diagram of the first version of the proofreader. Components depicted in white are separate TinyMCE modules. The horizontal axis suggests time.



3 The JavaScript-first approach

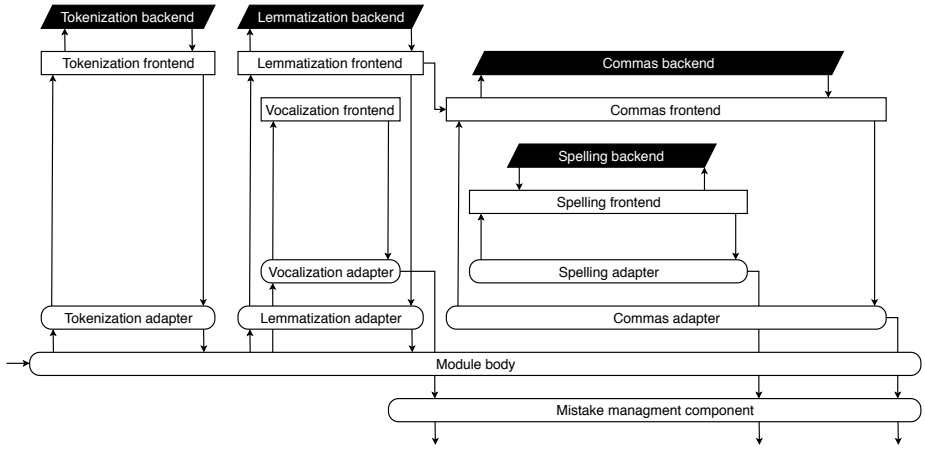
Learning from the previous mistakes, the second version, although created as separate JavaScript files, worked independently on the TinyMCE. This approach allowed to adapt the proofreader for other environments. Additionally, when used in one, the compatible versions of components could have been used and distributed as a single module (see fig. 2).

Because of the (at the time unknown) demand to use the proofreader in other environments than the browser applications, the development of the second version had to be cancelled as it would be unsustainable with these conditions.

4 The Python API approach

Currently in development is the third version of the proofreader created as an application programming interface written in Python. This provides a unified access point to be used in browser-based and non-browser based applications alike. The API-based approach also creates natural way to test the proofreader with large portion of texts as it does provide single machine-readable output. Along with the new approach, new features are presented as well as solutions to the existing problems.

Fig. 2: The workflow diagram of the second version of the proofreader. Elements depicted in white are components of the single module (where *frontend* means universal interface and *adapter* adaptation of the universal data to the TinyMCE specific environment). The horizontal axis suggests time.



4.1 Alternative tagger

The new version of the proofreader includes the alternative option to lemmatisation and tagging – *MorphoDiTa*[6] – and currently is being tested as an alternative to the combination of *majka* (lemmatiser and tagger)[8] and *DESAMB*[7] (disambiguator) which was used in previous versions.

The main problem with implementation is that the tagsets used by majka and MorphoDiTa differ in formal properties (attributive vs positional) as well as in encoded information. It follows that these two tagsets are not mutually convertible without losing some of the information provided. Conversion, however, has to be done since rulesets created for majority of proofreading components are dependent on majka's attributive tags. Similarly, some of the components need to have the source text separated into sentences. This information is not provided by MorphoDiTa. For the testing purposes, I used majka to help with sentence limits knowing it will need to be altered in the future to achieve better performance. A secondary problem is the attachment of additional information to the lemma such as lemma category (link to the PDT – Prague dependency treebank); thus the final output is for example `jet-1_^(pohybovat_se,_ne_však_chůzí)` instead of simple `jet`. This was resolved with a simple regular expression.

4.2 Asynchronicity and performance

Due to the single-query nature of the API, it is not possible to have mistakes displayed gradually depending on the time different components finish their

processing, but there has to be single output. However, an advantage of this approach is that there is higher pressure to make individual proofreading components faster. Besides, although the output can be only singular, the idea of asynchronicity and parallel processing of the output was preserved using the *asyncio* module, which was included into Python standard library in version 3.4.

Performance-wise there is a higher focus on the inner workings of Python, to achieve better speed. For example information about tagged words are stored as *named tuple* (from the *collections* module of the Python standard library) instead of widely-used *dictionary* keeping the data smaller memory-wise, but preserving the same readability.[5] See the following example for the sentence *Jak se máš?*:

```
[TaggedWord(word='Jak', lemma='jak', tag='k6eAd1', type='WORD'),
 TaggedWord(word=' ', lemma='', tag='', type='WHITESPACE'),
 TaggedWord(word='se', lemma='se', tag='k3xPyFc4', type='WORD'),
 TaggedWord(word=' ', lemma='', tag='', type='WHITESPACE'),
 TaggedWord(word='máš', lemma='mít',
            tag='k5eAaImIp2nS', type='WORD'),
 TaggedWord(word='?', lemma='?',
            tag='kIx.', type='MULTICHAR_PUNCTUATION')]
```

To keep the way to further optimisation open, the crucial components, such as the mistake manager, are wrapped inside the class, providing consistent output despite alterations of their inner workings.

4.3 Updates on proofreading components

As the structure of the proofreader changed heavily from the first version, the proofreading components had to be reworked as well. As the majority of these components used SET analyser as part of their doing, the adaptation was a relatively simple alteration of the backend part of the original component. Some of the components were updated with improved rulesets to provide better results. However, as adapting these components is currently ongoing or recently finished, there are no complex testing results to be published yet. There are also new components; for example, the component focused on detection of non-grammatical constructions such as attraction or blending errors.

5 Conclusion

This paper follows the current status of the proofreading tool developed at Masaryk University as well as the overall progress made during the recent year following the previous description of the project in *Towards the New Czech Grammar-checker*. Multiple issues, such as independence on the specific text processor, were resolved already with other planned to be resolved in the proximate future. Although the development brings many obstacles, overcoming them is necessary to bring the project to the successful end.

Acknowledgements This work was supported by the project of specific research *Čeština v jednotě synchronie a diachronie* (Czech language in unity of synchrony and diachrony; project no. MUNI/A/1061/2018) and by the Technology Agency of the Czech Republic under the project TL02000146.

References

1. Kovář, V., Horák, A., Jakubíček, M.: Syntactic Analysis Using Finite Patterns: A New Parsing System for Czech. In: Human Language Technology. Challenges for Computer Science and Linguistics. pp. 161–171. Springer, Berlin/Heidelberg (2011)
2. Mrkývka, V.: Towards the New Czech Grammar-checker. In: Horák, A., Rychlý, P., Rambousek, A. (eds.) Proceedings of the Twelfth Workshop on Recent Advances in Slavonic Natural Languages Processing, RASLAN 2018. pp. 3–8. Masaryk University, Brno (2009)
3. Novotná, M., Masopustová, M.: Using Syntax Analyser SET as a Grammar Checker for Czech. In: Horák, A., Rychlý, P., Rambousek, A. (eds.) Proceedings of the Twelfth Workshop on Recent Advances in Slavonic Natural Languages Processing, RASLAN 2018. pp. 9–14. Masaryk University, Brno (2009)
4. Petkevič, V.: Kontrola české gramatiky (český grammar checker). Studie z aplikované lingvistiky-Studies in Applied Linguistics **5**(2), 48–66 (2014)
5. Ramalho, L.: Fluent Python, p. 96. O'Reilly, Sebastopol (2015)
6. Straková, J., Straka, M., Hajič, J.: Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations. pp. 13–18. Association for Computational Linguistics, Baltimore, Maryland (June 2014), `http://www.aclweb.org/anthology/P/P14/P14-5003.pdf`
7. Šmerk, P.: Towards morphological disambiguation of Czech. Ph.D. thesis, Masaryk University, Brno (2007)
8. Šmerk, P.: Fast Morphological Analysis of Czech. In: Sojka, P., Horák, A. (eds.) Proceedings of Third Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2009. pp. 6–9. Masaryk University, Brno (2009)