Natural Language Engineering (2021), **27**, pp. 511–518
doi:10.1017/S1351324921000164

CAMBRIDGE UNIVERSITY PRESS

INDUSTRY WATCH

The automated writing assistance landscape in 2021

Robert Dale [1,*] and Jette Viethen [2]

[1]Language Technology Group and [2]Macquarie University
*Corresponding author. E-mail: rdale@language-technology.com

Abstract

Automated writing assistance – a category that encompasses a variety of computer-based tools that help with writing – has been around in one form or another for 60 years, although it's always been a relatively minor part of the NLP landscape. But the category has been given a substantial boost from recent advances in deep learning. We review some history, look at where things stand today, and consider where things might be going.

1. Introduction

This column has looked at commercially available tools for writing assistance – more specifically, grammar checkers and related technologies – twice before. In Dale (2004), we suggested that Microsoft's incorporation of grammar checking into MS Word, its near-ubiquitous word processing platform, was stifling innovation in the space. By bundling what was then a state-of-the-art approach to grammar checking with the market-leading desktop office software, effectively for free, it seemed that Microsoft had set the barrier to entry for third parties insurmountably high.

Then, 5 years ago, we surveyed how a number of new entrants had nonetheless managed to gain a foothold in the writing assistance marketplace (Dale 2016). We singled out Grammarly, founded in 2009, for particular attention, not least because of the significant visibility it had achieved through its somewhat relentless online marketing program.

Today, Grammarly is the clear market leader in tools whose primary function is writing assistance. In 2019, the company received US$90m in funding, which is a pretty big number for an NLP company;[a] and Grammarly now claims 30 million daily users.[b] That's nothing, of course, in comparison to Word's market penetration – even in 2016, Microsoft claimed there were 1.2 billion MS Office users.[c] But it's hard to find numbers on how many people actually make use of Word's grammar checker, and much easier to find anecdotal evidence that many either ignore its recommendations or switch it off. You'll also find plenty of blog posts claiming Grammarly's checker is better than Microsoft's, but few that argue the opposite.

Recent advances in deep learning look set to rearrange the automated writing assistance space once again, so now seems like a good time to take stock. Below, we review some history, look at where things stand today, and consider where they might go in the future.

2. The traditional landscape

Automated writing assistance, from the perspective of natural language processing, has traditionally consisted of three distinct capabilities whose purpose is to help authors address deficiencies in

[a]https://techcrunch.com/2019/10/10/grammarly-raises-90m-at-over-1b-valuation-for-its-ai-based-grammar-and-writing-tools/
[b]https://www.grammarly.com/about
[c]https://www.windowscentral.com/there-are-now-12-billion-office-users-60-million-office-365-commercial-customers

their writing: spell checking, grammar checking and style checking.[d] As we'll use those terms here, **spell checking** is about making sure that the sequence of characters that you type for the word you have in mind correspond to those that are generally accepted to constitute that word; **grammar checking** is about making sure that the sequences of words you type are consistent with the accepted rules of syntax in the language in which you're writing; and **style checking** is about making sure that, beyond correctness of spelling and grammar, the otherwise free choices you make in selecting how you say what you want to say are appropriate for your intended communicative purpose and target audience.[e]

2.1 Spell checking

Perhaps not surprisingly, spell checking has been around the longest, with the earliest instance of a spelling checker being developed a full 60 years ago by Les Earnest at Stanford back in 1961 (Earnest 2011).

Early approaches relied on lists of valid words, with any word in a text that was absent from such a list being considered as a potential spelling error. Apart from the obvious problem of long-tail vocabularies making the construction of such lists a seemingly endless task with diminishing returns, there was also the problem of 'real word' errors, where a word is misspelled as another valid word. It wasn't until the late 1980s and early 1990s that work on statistical language models provided a solution here, with *n*-gram probabilities being used to detect words that seemed unlikely in context.

Today, variations on these approaches are now used in, for example, Microsoft Word and Google Docs. Microsoft added contextual spelling checking into Word in 2007:[f] this initially appeared labelled as 'Use contextual spelling' in the settings panel, but has since been renamed as 'Frequently confused words'. In 2009, Google incorporated a context-sensitive spelling checker using a massive web-derived language model into the now-defunct Wave product (Whitelaw et al 2009).[g]

2.2 Style checking

In the late 1970s, Lorinda Cherry and Nina McDonald at Bell Labs developed the Unix Writer's Workbench, a suite of programs that aimed to help with a wide range of writing issues, many of which fall under the heading of style (Macdonald 1983). These programs were generally based on simple character string pattern matching, and included tools for the detection of duplicate words and split infinitives, and the identification of jargon, sexist language and other dispreferred words and phrases.

The Unix WWB gained a strong foothold in academia by virtue of being bundled with the operating system of choice for many educational institutions. Outside the academic sector, a number of commercial entities were quick to replicate the key functionalities of the WWB on the personal computer platforms that were coming onto the market in the early 1980s. The most well known of these products was Grammatik, heralded as the first grammar-checking program on PCs.

We might baulk at calling these pattern-matching approaches 'grammar checking' today, but it's important to distinguish the categorisation of the phenomena being addressed from the

[d] You might consider the presence in an editor of a built-in dictionary or thesaurus, or organisational functionalities like MS Word's outline mode to be forms of writing assistance, but these don't involve what we would consider to be the processing of language, so we deem them out of scope here.

[e] There is, of course, an important technical distinction between *detecting* an error of a given type and *proposing a correction* for that error; however, the present discussion isn't particularly impacted by this distinction.

[f] https://www.wsj.com/articles/SB116786111022966326.

[g] https://www.youtube.com/watch?v=Sx3Fpw0XCXk.

https://doi.org/10.1017/S1351324921000164 Published online by Cambridge University Press

techniques used to identify or correct those phenomena. With the aid of simple regular expressions, a range of common grammatical errors can be captured; and if a sentence structure is too complex for a regular expression to detect a potential issue, a prior check for too-long sentences might just bounce that back to the author before the simple approach to grammar reveals its shortcomings.

Other software packages that appeared around the same time and provided similar capabilities were RightWriter, Punctuation & Style, Correct Grammar and PowerEdit; thanks to the Internet Archive, you can find a fascinating comparative review of a number of these packages in a 1991 issue of *InfoWorld*.[h]

What's interesting here is that so much can be achieved even with these simple techniques. You can't solve all spelling, grammar and stylistic problems with simple pattern matching, but you can still do a lot; and so, as we'll see further below, these techniques still have a prominent place in today's writing assistance landscape.

2.3 Grammar checking

As just noted, regular expressions can help you identify some grammatical errors. But faced with a long subject noun phrase that contains a doubly embedded relative clause, it's almost certain that a simple pattern-matching approach will look at the wrong tokens when it attempts to check for subject–verb number agreement. The predominant view in the 1980s and early 1990s was that to do grammar checking properly, you needed a comprehensive broad-coverage phrase structure grammar, along with a mechanism for handling word sequences not recognised as correct by that grammar.

There's an extensive research literature on grammar checking, much of it from the 1980s when grammars and parsing seemed to absorb the bulk of the NLP research community's energy. It's not clear that much of that research found its way into commercial products, however, with one very prominent exception. What at the time was considered 'real' grammar checking came to the fore with the work of Karen Jensen and George Heidorn, first at IBM with their development of the EPISTLE system (Heidorn et al 1982), and subsequently at Microsoft, where they were responsible for the first widely available grammar checker based on full parsing of the text being analysed, released as a component of Microsoft Word in 1997.

In the days before we cottoned on to the idea of learning a grammar from a treebank, developing such a broad-coverage grammatical resource was a mammoth undertaking, and very few companies appear to have attempted it. Later versions of Grammatik, last seen as an inclusion in WordPerfect, appeared to embody some degree of syntactic parsing; and prior to Jensen and Heidorn's work, Microsoft Word made use of a product called CorrecText, which was claimed to use a full-blown grammatical analysis (Dobrin 1990). The aforementioned Grammarly also appears to use some form of syntactic analysis. But the only publicly well-documented commercial-grade grammar-based syntax checker remains the system used in Word (Heidorn 2000).

3. Today's landscape: old meets new

So what does the world of commercially available tools for automated writing assistance look like today? To get a sense of this, we surveyed over 50 currently available commercial tools for writing assistance, leading us to identify three broad categories:

(1) Pattern-matching style checkers: although almost all the programs developed in the 1980s have disappeared, their spirit lives on in a number of successful newer applications.

[h] https://archive.org/details/bub_gb_dT0EAAAAMBAJ/page/n67/mode/2up

https://doi.org/10.1017/S1351324921000164 Published online by Cambridge University Press

(2) Language-model-driven rewriting tools: this new category provides an alternative to the traditional approaches to correcting text.

(3) Text generation tools: capable of creating texts of various lengths from a user-provided prompt, these represent a paradigm shift from computer-as-editor to computer-as-author.

We pick out what we consider to be the more interesting of these applications below.

3.1 Pattern-matching style checkers

The particular style-checking applications that were around in the 1980s have all but disappeared. RightWriter is still represented by a website (http://www.right-writer.com) that serves as a great reminder of what 1980s websites looked like, but if you click on the link to purchase the software, nothing happens; the site's only functioning external link is to an essay mill, and one suspects the site has been hijacked just to host that link.

None of the other tools from the 1980s are still visible on the web, with one interesting exception: Stylewriter (https://www.editorsoftware.com), which first appeared in 1989, still seems to be going strong. Its website predates the slick and simple design of modern website templates, and the business model similarly comes from an earlier age: this is an application you buy via a one-time purchase (albeit with optional annual upgrades and support at additional cost), as opposed to renting on a monthly basis via the now ubiquitous SaaS model.

But style checking is a space where age and experience may count for something. At the heart of these applications lie manually created rules that detect quite specific patterns in text, suggest equally specific corrections and often provide detailed manually curated explanations of what's at issue. Other things being equal, the longer you've been around, the more such rules your software will have accumulated. StyleWriter checks for 50,000 words and phrases, as well as making use of a list of 200,000 words graded by difficulty; and the website claims that 1000 new checks are added each year. The application also includes a regular expression language that you can use to add new pattern-matching rules if your favourite bugbear isn't already covered.

What's more interesting here is how the same old-school technology serves as the basis for a crop of much newer applications. Of these, the Hemingway Editor (https://hemingwayapp.com; founded 2014) looks to be the simplest. This makes use of a straightforward and uncluttered interface to recommend changes that make your text 'bold and clear'; its scope is limited to detecting the use of adverbs and passives, suggesting simpler forms of expression, and objecting to hard-to-read sentences. There's no pretense that the tool does grammar checking; the focus is on simplifying the text through detecting specific patterns that can be removed or replaced.

Many of the applications in this space target professional writers, and consequently often provide support for sharing style settings across teams of users. Lingofy (https://www.lingofy.com) and Linguix (https://linguix.com) both fall into this category; in each case you can use already existing rule sets for style checking, or augment these with your own rules. Linguix also claims to check grammar, but this capability appears limited to errors that can be detected by simple pattern matching.

PerfectIt (https://intelligentediting.com; founded 2009) focusses particularly on consistency of usage around the more mechanical aspects of house style, such as the formatting of alphanumeric expressions, token-internal punctuation and geographic spelling preferences; it also provides extensive support for tailoring style sheets to specific needs. StyleGuard (https://styleguard.com) appears to be an implementation of the Associated Press style guide, with 'more than 25,000 style rules'; again, all the examples provided appear to be implementable via pattern matching.

WordRake (https://www.wordrake.com; founded 2012), which appears to be aimed particularly at the legal profession, again focusses on editing for clarity and brevity; it distinguishes itself from many other applications by emphasising a commitment to privacy, by virtue of being a

https://doi.org/10.1017/S1351324921000164 Published online by Cambridge University Press

desktop implementation that doesn't send any data or documents to the cloud. The application is backed by a number of remarkably detailed patents, although these appear to describe rules that are very similar to those that can be captured using the pattern-matching languages offered by a number of other vendors.

ProWritingAid (https://prowritingaid.com; founded 2012), which we also include in the category of applications described in the following section, exposes what appears to be the most sophisticated language for writing your own rules, encompassing both part-of-speech tags and inflectional morphology; this puts it on a par with the open-source tools Language Tool (https://dev.languagetool.org; Naber (2003)) and After the Deadline (https://www.afterthedeadline.com; Mudge (2010)).

3.2 Language models in text rewriting

Statistical language models of all kinds have found their way into writing assistance tools. In 2019, Google introduced enhanced grammar-correction tools in Google Docs, adopting a now-common approach of viewing grammatical error correction as a machine translation task, and using neural MT techniques to detect and correct errors in text.[i] Also in 2019, Grammarly published research that described how they used a Transformer language model for grammatical error correction.[j]

But the availability of large language models also supports new writing assistance tasks that were previously not feasible. Once we allow a language model to propose an alternative to a provided text, the detection and correction of errors or infelicities becomes a special case of the more general task of text rewriting. The ability to offer alternative renderings of existing sentences, regardless of whether they contain errors, has become the new must-have feature in writing tools; and so there are now many applications which combine error detection with sentence-level rewriting.

It's rare for vendors to be explicit about the precise nature of the technologies they use, other than the directional hint that is provided by a claim to be using some form of AI. As far as we can tell, the following make use of large language models to correct or transform text:

- Ginger (https://www.gingersoftware.com; founded 2007) appears to make use of fairly sophisticated syntactic analysis, but also makes use of an 'AI-based' sentence rephraser.
- Gramara (https://gramara.com) calls itself an 'AI-powered grammar checker'; it lets you specify the tone of its rewrites, and also offers to translate your text.
- Outwrite Pro (https://www.outwrite.com; founded as GradeProof in 2015) places a heavy emphasis on its full-sentence paraphrase capabilities.
- Wordtune (https://www.wordtune.com) rewrites a provided text in either casual or formal style, and can shorten or expand the originally provided text.

As well as rewriting text, these applications generally also detect specific errors, providing an interface much like that offered by the more traditional checkers, whereby errors signalled in the text are associated with pop-ups that categorise the error type, provide a potential replacement and allow for the corresponding rule to be ignored or switched off. It's usually not clear whether these functionalities are provided via simple rules or something more sophisticated, although the latter is certainly a possibility: according to its creators, the earlier-mentioned ProWritingAid combines both a pattern-matching approach to detect common problems, and an approach that

[i] https://cloud.google.com/blog/products/productivity-collaboration/using-neural-machine-translation-to-correct-grammatical-in-google-docs

[j] https://www.grammarly.com/blog/engineering/under-the-hood-at-grammarly-leveraging-transformer-language-models-for-grammatical-error-correction/

https://doi.org/10.1017/S1351324921000164 Published online by Cambridge University Press

uses a deep-learning model to create corrected texts that are then labelled with error types using a more conventional classification algorithm.

3.3 From editor to author

Whether we're talking about spelling, grammar or style, automated writing assistance has been traditionally concerned with correcting or improving text written by a human author. The impressive ability of large language models like OpenAI's GPT-3 to generate plausible and convincing texts on any topic opens the door to a new set of capabilities, where responsibility for the content of a text is shared between human and machine.

The first inkling of this possibility was the 2015 release of Google's Smart Reply functionality, whereby for certain kinds of received mails, Gmail would offer a set of simple one-click response messages whose content was determined on the basis of the received mail.[k] This was effectively a tightly constrained version of something more sophisticated to come: Google's 2018 Smart Compose feature, which interactively offered sentence completion suggestions as you type an email, taking into account not only what you've typed already, but also the email you're replying to and the subject line of the email.[l]

These are instances of what we have elsewhere referred to as 'short-leash' generation (Dale 2020): the machine-generated text isn't allowed to stray too far, sufficiently constrained by what's gone before that there's a very good chance that it's an accurate prediction of what you might have said anyway.

But the release just over a year ago of Open AI's GPT-3 API has significantly changed what's feasible here. The plausibility of texts created by earlier and smaller language models was not great, particularly as they got longer; the first few sentences might seem ok, but thereafter the outputs would rapidly degenerate into nonsense. GPT-3 is still very capable of generating nonsense, but on the whole it's more plausible nonsense; and with appropriate fine tuning and prompting, the texts it generates can be eerily convincing.

Consequently, there are now literally dozens of applications that will help you write by writing for you, given a brief prompt. Table 1 lists a sample; by the time you read this, some of these may have disappeared, to be replaced by others. Most of these allow you to sign up for a free-access trial, which is worth doing to get an idea of what they can do. The outputs are often weak, again especially so for longer texts, but occasionally you will be genuinely impressed; and even the weaker results may provide a source of inspiration if you have writer's block.

Each application supports a range of use cases, sometimes referred to as templates or tasks: amongst the most common are shorter texts like headline generation, product descriptions and advertising copy, but quite a few also support longer texts and less common use cases. The applications vary in their interfaces and the degree of control they provide over the generated output; the quality of the results in each case will depend to some extent on the tuning carried out by the developers for their specific use cases.

4. Where to from here?

Some observations on the set of capabilities that we characterised earlier as defining the scope of automated writing assistance:

- **Spell checking is a commodity.** That's not to say it's a solved problem – no spelling checker will catch 100% of spelling errors – and neither does it mean that every spell checker you come across provides state-of-the-art performance; but pretty much any platform that

[k] https://research.google/pubs/pub45189/
[l] https://ai.googleblog.com/2018/05/smart-compose-using-neural-networks-to.html

https://doi.org/10.1017/S1351324921000164 Published online by Cambridge University Press

Table 1. Some GPT-3 content writers

Tool	Use cases
AI Writer	Given a title, writes an article; also rewrites existing text
Anyword.com	Social media and marketing content, email subject lines, product descriptions
Articoolo	Writes an article or essay given a topic; also rewrites existing text
ContentBot	Blog topics, blog post content, ad copy, brand names
Conversion.ai	Social media and marketing content, blog posts
CopyAI	Around 50 different narrowly targetted use cases
Copy Shark	Ad copy, product descriptions, sales copy, blog paragraphs, video scripts
Copysmith	Ads, product descriptions, emails
Craftly	Around 10 use cases including mission statements
Nichess	Around 30 use cases, including poetry and real estate descriptions
Rytr	25+ use cases
Shortly AI	Writing assistance; rewriting, expansion and shortening
Snazzy AI	20+ use cases
Text Cortex	Product descriptions, social media ads, blog posts
Writesonic	Ads, blogs, landing pages, product descriptions

supports the editing of text today incorporates a spelling checker. For most authors, it seems sufficient to know that there's some kind of spelling safety net, even if there might be holes in that net. Reassured by the first red squiggly underline that pops up, we assume that the most common or egregious errors will be caught, and take the risk that those errors that might be missed will have a low probability of occurrence.

- **Attitudes to grammar checking vary.** For some vendors, the battleground has moved to the more general task of 'improving writing': they see the grammar checking war as having been won by the incumbents, and don't consider any further fighting worth the effort. These vendors effectively pitch their solutions as applications you use once grammar checking has been carried out via Word, Grammarly or whatever; implying that grammar checking is effectively routine, they advertise their focus as addressing problems that are *beyond* grammar. Other vendors candidly admit that they don't attempt grammar checking because it's too hard, perhaps implying that it can't really be done at all. And then there are those that claim to use AI to catch grammar errors that the more mainstream products can't.
- **Style never goes out of fashion.** As we've seen, applications that use 1980s technology to address stylistic issues are still going strong. And the new wave of applications that use large language models to rewrite texts are arguably all about style: given an original sentence and a rewritten version that conveys the same content, you can often characterise the difference as being purely stylistic. Large language models may encourage us to think about style in much more holistic terms as a property of entire texts; perhaps the 1980s approach of characterising style in terms of the appropriateness or otherwise of specific word sequences is a case of not seeing the wood for the trees.

To maintain its standing in the face of so many new applications in the space, Grammarly has long de-emphasised a specific focus on grammar checking (shame about the name, eh?);

https://doi.org/10.1017/S1351324921000164 Published online by Cambridge University Press

at the time of writing, its current tagline was 'Compose bold, clear, mistake-free writing with Grammarly's AI-powered writing assistant'. Microsoft similarly has repackaged and extended its language proofing capabilities as Microsoft Editor, a tool which incorporates both text prediction and rewriting via a plug-in, which also extends its reach beyond the standard Microsoft platforms.

It remains to be seen whether the more traditional style-checking solutions will continue to thrive, faced with the rapid developments in the large language model space. It's important not to downplay the significant assets many of these companies possess: apart from the large, carefully curated rule sets they have accumulated over time, complete with sometimes detailed explanations that so far are beyond the wit of language models, there's also the ecosystems they have developed around their products by providing extensive additional support for the writing task via email newsletters and blog posts that address specific writing problems or conundrums. Not only do these ancillary resources provide some stickiness that might promote customer loyalty, they also provide a convenient route for delivering support for writing issues that are not or cannot be handled by the software itself, partially compensating for the software's limitations. But it's more likely that to survive, these players will have to adapt, incorporating the newer technologies to keep up with the competition; ProWritingAid's hybrid approach is a good example here.

But the big shift is the transition from tools that help with editing to tools that help with authoring. It's conceivable that, in 5 years' time, no automated writing assistance tool will be considered complete without a functionality that finishes your sentences, writes you out of tight corners and fills in background paragraphs while you doze at the keyboard. And given Microsoft's exclusive licencing deal with OpenAI in regard to GPT-3,[m] it won't be a surprise if, before too long, we see some of these capabilities as yet another item on Microsoft Word's ribbon menu.

References

Dale, **R.** (2004). Industry Watch. *Natural Language Engineering* **10**, 91–94.

Dale, **R.** (2016). Checking in on grammar checking. *Natural Language Engineering* **22**, 491–495.

Dale, **R.** (2020). Natural language generation: The commercial state of the art in 2020. *Natural Language Engineering* **26**, 481–487.

Dobrin, **D.N.** (1990). A New Grammar Checker. *Computers and the Humanities* **24**, 67–80.

Earnest, **L.** (2011). The first three spelling checkers. Archived at https://web.archive.org/web/20121022091418/, http://www.stanford.edu/learnest/spelling.pdf.

Heidorn, **G.** (2000). Intelligent writing assistance. In Dale, R., Moisl H. and Somers H. (eds), *A Handbook of Natural Language Processing: Techniques and Applications for the Processing of Language as Text*, pp. 181–207. New York: Marcel Dekker.

Heidorn, **G.**, **Jensen**, **K.**, **Miller**, **L.**, **Byrd**, **R. and Chodorow**, **M.** (1982). The EPISTLE text critiquing system. *IBM Systems Journal* **21**, 305–326.

Macdonald, **N.H.** (1983). The UNIX writers workbench software: rationale and design. *Bell System Technical Journal* **62**, 1891–1908.

Mays, **E.**, **Damerau**, **F.J. and Mercer**, **R.L.** (1991). Context based spelling correction. *Information Processing and Management* **27**, 517–522.

Mudge, **R.** (2010). The Design of a Proofreading Software Service. Pages 24–32 in *Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics and Writing: Writing Processes and Authoring Aids*, June, Los Angeles, CA, USA.

Naber, **D.** (2003). A Rule-Based Style and Grammar Checker, Diploma Thesis, University of Bielefeld.

Whitelaw, **C.**, **Hutchinson**, **B.**, **Chung**, **G. and Ellis**, **G.** (2009). Using the Web for language independent spellchecking and autocorrection. Pages 890–899 in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, August, Singapore.

[m] https://blogs.microsoft.com/blog/2020/09/22/microsoft-teams-up-with-openai-to-exclusively-license-gpt-3-language-model/

Cite this article: Dale R and Viethen J (2021). The automated writing assistance landscape in 2021. *Natural Language Engineering* **27**, 511–518. https://doi.org/10.1017/S1351324921000164