# Envisioning the Next-Gen Document Reader

Catherine Yeh[1], Nedim Lipka[2] and Franck Dernoncourt[2]

[1]Harvard University
[2]Adobe Research
catherineyeh@g.harvard.edu, lipka@adobe.com, franck.dernoncourt@adobe.com

**Abstract**

People read digital documents on a daily basis to share, exchange, and understand information in electronic settings. However, current document readers create a static, isolated reading experience, which does not support users' goals of gaining more knowledge and performing additional tasks through document interaction. In this work, we present our vision for the next-gen document reader that strives to enhance user understanding and create a more connected, trustworthy information experience. We describe 18 NLP-powered features to add to existing document readers and propose a novel plug-in marketplace that allows users to further customize their reading experience, as demonstrated through 3 exploratory UI prototypes available at: github.com/catherinesyeh/nextgen-prototypes.

**Keywords**

Natural language processing, document readers, UI prototypes

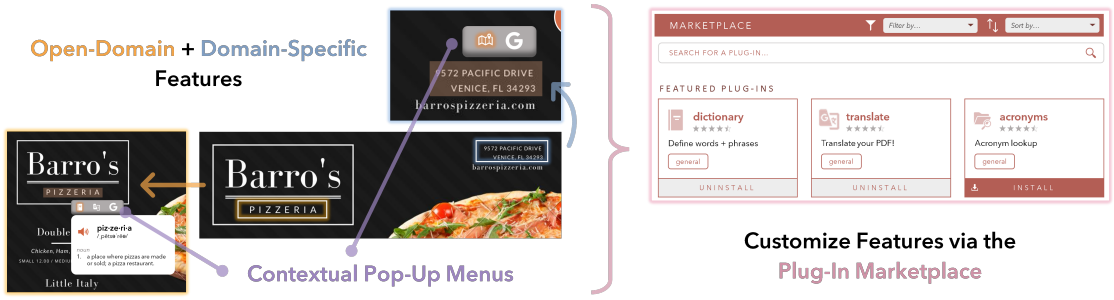


Figure 1: Overview of our vision for the next-gen document reader. We propose transforming static, isolated documents into a trustworthy, connected, and interactive sources of information through open-domain and domain-specific features, contextual plug-in pop-up menus, and a centralized plug-in marketplace. (Menu source: PDF Archive website)

## 1. Introduction

Digital documents (e.g., portable document format (PDF) files or Word documents) are a popular format for sharing, exchanging, and understanding information in electronic settings. Reading such documents is an integral part of countless people's daily routines, and many choose to engage with these files through document readers such as *Adobe Acrobat*, *Foxit*, and *Sumatra PDF*. However, with these current applications, document reading can feel relatively static and isolated, as the reading experience is usually confined to within the document reader itself. Additionally, there is typically not much interaction between the user and the information they are reading when scrolling through a digital document.

This presents a problem as documents themselves are usually not the end goal for users. Rather, they represent a starting point for people to gain more knowledge or perform additional actions. Thus, in this work, we present our **vision for the next-gen document reader** that strives to better support users in achieving their goals through harnessing the power of natural language processing (NLP). We design this next-gen document reader to 1) enhance user understanding of digital files and 2) transform currently static, isolated documents into connected, trustworthy, and interactive sources of information.

The key contributions of our work include:

- A set of proposed NLP-powered plug-ins to add to existing document readers toward enhancing

*SDU @ AAAI '23: The Third AAAI Workshop on Scientific Document Understanding, February 14, 2023, Washington, DC*

human-document interaction, including 12 **open-domain** and 6 **domain-specific** features.
- A preliminary vision for a centralized **plug-in marketplace** that would allow further customization of the user experience in document readers and feature development to be outsourced.
- 3 exploratory **UI prototypes** illustrating a subset of features and the plug-in marketplace proposed for the next-gen document reader (github.com/catherinesyeh/nextgen-prototypes).

## 2. Related Work

While older document readers such as *Adobe Acrobat*, *Foxit*, and *Sumatra PDF* tend to only support static, in-document features, recent NLP efforts are beginning to explore the possibility of creating a more connected, trustworthy information experience for users.

For example, *ScholarPhi* [1] strives to improve the readability of scientific papers by creating an augmented reading interface with features such as position-sensitive definitions, a decluttering filter, and an automatically generated glossary for the important terms and symbols. Similarly, *Paper Plain* [2] is an interactive interface that aims to make medical research papers more accessible with its definition feature, section gists, and Q & A passages. *Scim* [3] is another AI-augmented document reader that helps researchers skim scientific papers by automatically identifying, classifying, and highlighting salient sentences.

*Sioyek* [4], a document viewer designed for reading technical books and research papers, has some interesting features such as smart jump for references and figures, searchable bookmarks, and portals to display linked information (e.g., figures and formulas) in a separate window. *Explainpaper* is a novel AI-powered reading interface for reading academic papers as well, offering live explanations to users upon highlighting sections of text and an interactive Q & A feature. However, these works are currently very limited in their features and scope.

## 3. Vision

Our vision for the next-gen document reader includes the following components as illustrated in Figure 1:

- A set of **open-domain** features that can enhance the document reading experience for various document types,
- A set of more **domain-specific** features, and
- A centralized **plug-in marketplace** that would allow users to further customize document readers with additional features.

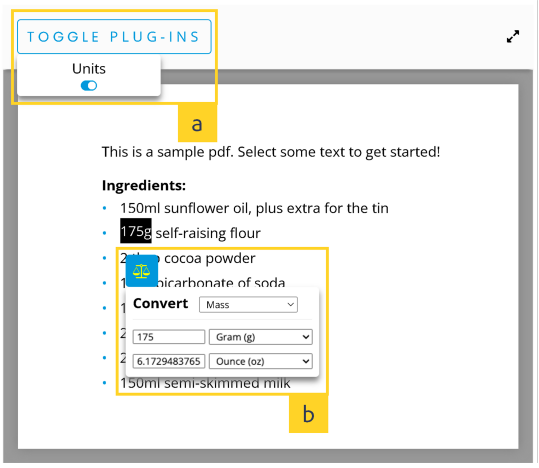


**Figure 2:** UI prototype demonstrating our contextual pop-up menu concept. Users can a) toggle installed plug-ins onoff and when active, b) the corresponding plug-in tooltip only appears if relevant text is selected. *(Recipe: BBC Good Food)*

Throughout this paper, we use plug-in and feature synonymously to mean any software add-on that serves to extend the core functionality of a static document reader. Once a plug-in is installed, it could be toggled on/off by users and when active, the plug-in could be accessed through a **contextual pop-up menu**. Figure 2 illustrates how this type of menu could work with a sample PDF cake recipe. In this UI prototype, the unit conversion plug-in is toggled on, but the corresponding tooltip icon only appears if relevant text is selected (i.e., text containing numerical values). Ideally, the document reader would also automatically identify the correct unit of measurement selected by the user and auto-populate this information into the pop-up conversion tool.

In the following sections, we provide more details about our proposed features and plug-in marketplace for the next-gen document reader.

### 3.1. Features

To begin the design process, we brainstormed features that would be helpful to add to static document readers such as *Acrobat* or *Foxit*, focusing on features that can leverage NLP. During this stage, we surveyed the literature and investigated existing plug-ins supported by newer document viewers [2, 3, 1] as described in the Related Work section. Some ideas were also contributed by peers and collaborators.

This process resulted in 26 potential feature suggestions, which we narrowed down to 18 based on feasibility of implementation. These ideas were then categorized by domain type, ultimately yielding 12 **open-domain**

**Table 1**
List of open-domain and domain-specific features proposed for the next-gen document reader

| **Open-Domain** | |
|---|---|
| Definitions | Equation Exporting |
| Acronyms/Abbreviations | Speed Reading |
| Unit Conversions | Sentiment Analysis |
| Translations | Form Auto-Fill |
| Spelling Changes | Scholar Notes |
| Table Copying | Shared Commenting |
| **Domain-Specific** | |
| Linkifying Known Entities | Smart Jumps |
| Linkifying Relevant Content | Citation Warnings |
| Action Tasks | Portals |

and 6 **domain-specific** features. A list of all proposed features is included in Table 1. Selected features are highlighted below using our second UI prototype of a PDF pizza restaurant menu (as previewed in Figure 1).

### 3.1.1. Open-Domain Features

When a single word is highlighted, document readers could show users potential **definitions** of the term (Figure 3), similar to [2, 1]. The displayed definitions could be retrieved from the document itself [5, 6], from online sources as in [2], or crowd-sourced.


**Figure 3:** Example definition feature

Similarly, the long forms of **acronyms/abbreviations** could be shown to the user via pop-up tool tips (Figure 4). If the acronym is defined elsewhere in the paper, we could take a similar approach to [7, 8, 9] for extracting definitions; otherwise, retrieving it from online sources is also possible. A list of key definitions and acronyms could be included at the beginning or the end of the document as well, following [10]. The definition of math symbols could also be extracted from the text [11, 12].

**Unit conversions** could be either a case-by-case basis (i.e., users highlight specific numbers to convert like definitions/acronyms) or document-level (i.e., document reader automatically converts all units at once). Figure 2

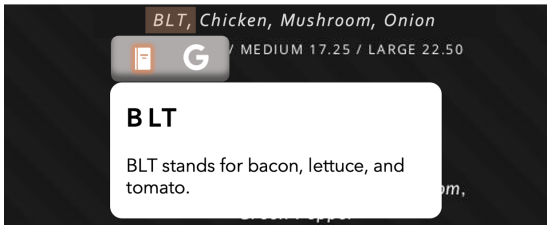


**Figure 4:** Example acronym feature

shows the former option, assuming that unit selection is embedded inside of the plug-in pop-up. Allowing users to select their unit of choice via the main document reader toolbar (see Figure 5) would be another possibility.

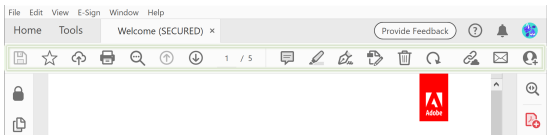


**Figure 5:** Main toolbar in Adobe Acrobat

As with unit conversions, there could be a toolbar option at the top of document readers that would allow users to choose the language they want to read the document in. Alternatively, **translations** could be performed on a more case-by-case basis. The former could be similar to the Google Translate browser extension. Figure 6 illustrates the latter option. Related to translation is the idea of automatically suggesting **spelling changes** based on the current document language. For example, the spellings in a document could automatically be converted from American to British English (e.g., color → colour).

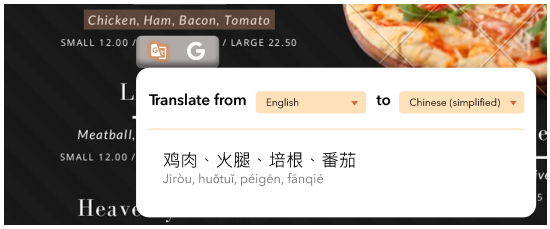


**Figure 6:** Example translation feature

Document readers could also provide users with the capability of directly copying tables from files like PDFs into Microsoft Word, Excel, Markdown, etc. to further manipulate, share or analyze. This **table copying** feature would reduce the need for hand transcribing and combining data from multiple tables in digital files. There could also be a selection hierarchy, allowing users to select

specific parts of a table (e.g., a single cell, row, column, etc.) or the whole table itself. A version of this feature is included in the *Adobe PDF Extract API*, but currently, tables can only be exported to CSV formats, so there is room for extension and it is still missing from all the PDF readers we surveyed. Similarly, an **equation exporting** plug-in could allow users to export math equations present in digital documents to their corresponding LaTeX formulas so they are directly editable. Implementing this feature would be possible using image-to-latex algorithms [13, 14, 15].

Another way to enhance the document reading experience could be including a **speed-reading** plug-in that would allow allow users to customize the speed at which they read text in document readers, similar to the service offered by *Spritz*. Additionally, a **sentiment analysis** feature could allow users to assess sentiment at the document level and potentially at the sentence level as well. Sentiment classification would be useful for a wide variety of document types, particularly when it is beneficial to understand a document's tone/attitude. Some approaches for document-level sentiment analysis have been proposed by [16, 17].

For certain documents such as history books, scientific papers, and poems, reading applications could also offer **scholar notes**. As an example, the document reader could include a critic's analysis of a text (e.g., in the sidebar) and/or their annotations throughout the file as comments that the user could view. The notes could be distributed via a marketplace, and some of them could be set as paid access only if monetization is of interest. Users may be willing to pay to get access to the meta-information given by a scholar in the field to better understand the text itself, its historical context, the equations, potential errors, the author's mindset at the time of the writing, and so on. A related feature idea is allowing users to leave **shared comments** in digital documents. For example, users could highlight a sentence or figure and then create a thread for further discussion (e.g., asking a question, offering clarification, etc.), which could open up in a sidebar.

These human-in-the-loop features could help make up for the imperfections and the limitations of other AI-powered document plug-ins. However, the main challenges with implementing such features would be moderating/filtering the user content and respecting users' privacy (e.g., we do not want a user to mistakenly post their comments as public if they did not intend too).

### 3.1.2. Domain-Specific Features

Document readers could also incorporate domain-specific features such as **linking text to known entities**. For example, addresses or business names could be automatically linked to Google Maps and phone numbers could be linked to an app/website for further action, as Google Chrome or Android currently does. The former is illustrated in Figure 7. Other ideas include linking protein names to the Protein Data Bank for biology documents, linking references to their Google Scholar entry in scholarly articles, or linking ticker symbols to their Yahoo Finance pages (e.g., finance.yahoo.com/quote/ADBE → ADBE) for finance documents.

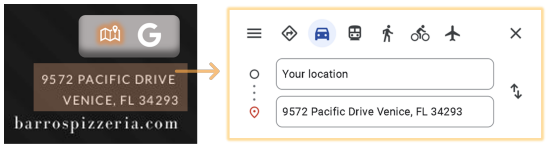


**Figure 7:** Example address linkifying feature

Similarly, document readers could **link text to relevant content**. This is a trickier task than linkifying known entities, but the required extrapolation may be feasible in certain cases. For instance, if a file is identified as a restaurant menu, the document reader could link to the corresponding Yelp or Google reviews page so users could see more pictures/reviews of different items (Figure 8). Or, if a movie title is identified inside a document, links to available movie times or streaming platforms could be generated. Another possibility would be searching selected keywords/phrases in a search engine or e-commerce website (e.g., Amazon, Alibaba, etc.) to see related products; *Axesso Amazon API* has implemented one such keyword search feature. Ultimately, this feature could be similar to how YouTube recommends products based on the videos a user watches.


**Figure 8:** Example restaurant linkifying feature

Additionally, we could identify and create **action tasks** that users could complete within the document reader itself. For example, if a date/deadline is identified in the text, users could be given the option to add it to their calendar. Similarly, if a payment is mentioned, users could have the option to pay directly inside of the document. Or, if there is language such as "You should notify X..." or "Please reach out to Y..." in a digital file, it might be helpful to give users the ability to send messages/e-mails from the document reader as well. In general, these tasks could be accessed via pop-up icons throughout the document, but there could also be an overall list on the sidebar, for instance.

For documents like academic papers or textbooks, a **smart jump** feature (terminology from *Sioyek*) could be offered that allows users to jump to any referenced figure or reference in the document, even if links are not explicitly provided. A similar feature has already been implemented by *Sioyek*. Currently, *Sioyek*'s smart jump feature automatically links references to their Google Scholar page as well, connecting to our idea for linkifying known entities described above. Another related idea would be including a **citation warning** feature that displays a warning to users when a citation in a document has been retracted [18, 19].

One last domain-specific feature would be **portals** (terminology from *Sioyek*). This feature would allow users to link figures to specific paragraph locations so they can view them simultaneously on a separate monitor/window, as implemented by *Sioyek's* portal feature (Figure 9). Such "portals" would enhance the reading experience by removing the need to scroll back and forth in a document to find the relevant figures for each section of text, which could especially be helpful for academic/scientific papers.

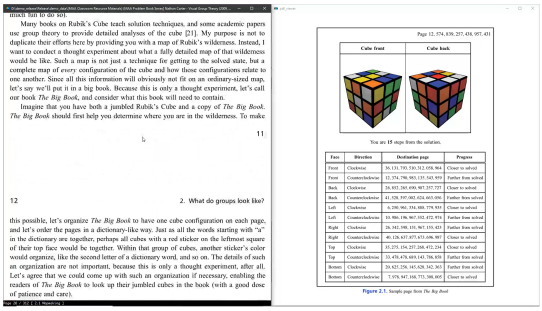

**Figure 9:** Example portal feature *(Source: Sioyek)*

## 3.2. Plug-in Marketplace

In the previous sections, we describe many potential plug-ins for the next-gen document reader. However, the average user will not need all these features when engaging with digital files. Thus, to allow users to further customize their document reading experience and choose which features they want to use, we propose the creation of a centralized **plug-in marketplace**. This way, document readers could come with a few default plug-ins (e.g., definitions/acronyms or other open-domain features that could be useful for most document types and users) and users could add more via the marketplace at any time.

Having a plug-in marketplace would also prevent document readers from growing excessively in terms of size and computational requirements. Instead, the user would individually decide which plug-ins to install and use, just like in virtually all modern code and text editors.

### 3.2.1. Key Features

An example of what such a marketplace might look like is illustrated in Figure 10. To our knowledge, there currently exist no plug-in marketplaces for document readers such as *Adobe Acrobat*, *Foxit*, and *Sumatra PDF*. Consequently, many features we include in our UI prototype are inspired by the marketplaces for integrated development environments (IDEs) like *Visual Studio*, *IntelliJ*, and *Eclipse* as well as marketplaces for text editors like *Emacs*, *Notepad++*, and *Sublime Text*.

On the main marketplace page (Figure 10), we envision a space where users can **discover** new features with the search bar or featured plug-ins section, which could highlight the newest or most popular plug-ins. Plug-ins could also be tagged (e.g., by domain) and reviewed to allow users to **filter** and **sort** the results. Inside the marketplace, users would have the option to **(un)install** plugins at any time.

Each user could also have their own "My Plugins" page inside the plug-in marketplace, as shown in Figure 11. On this page, users would again have the ability to search for, filter, sort, and uninstall plugins. In addition, users could **pin** their favorite plug-ins for easy access. Here, users would be able to globally **toggle plugins on/off** as well. There could also be an option to turn plugins on/off at a document level, illustrated in Figure 2a.

Another feature that could be added within the plug-in marketplace is a **feedback/feature request** page, where users could submit general feedback about the marketplace or propose new ideas for features to add. Along these lines, several IDEs/text editors include a community **forum** (e.g., eclipse.org/forums) where users can openly discuss topics and ask questions about different plug-ins, so implementing a similar discussion platform for document readers could also be valuable.

# 4. Future Work

This work represents a preliminary, exploratory vision for the next-gen document reader. The next steps include concretizing our ideas and assessing the viability of implementation. Specifically, we hope to conduct formal **user studies** to collect additional feedback on our vision, further hone the proposed designs, and better understand which features would be most useful to end-users. Through these user studies, we may also generate additional ideas for potential document reader plugins.

Further out in the future, we could also consider more **complex features**. For example, a filtering option would help readers focus on only the most relevant parts of the document, similar to the declutter feature from [2]. Similarly, fact-checking sentences and displaying a warning symbol next to text containing incorrect facts would be extremely valuable. Other complex features for future

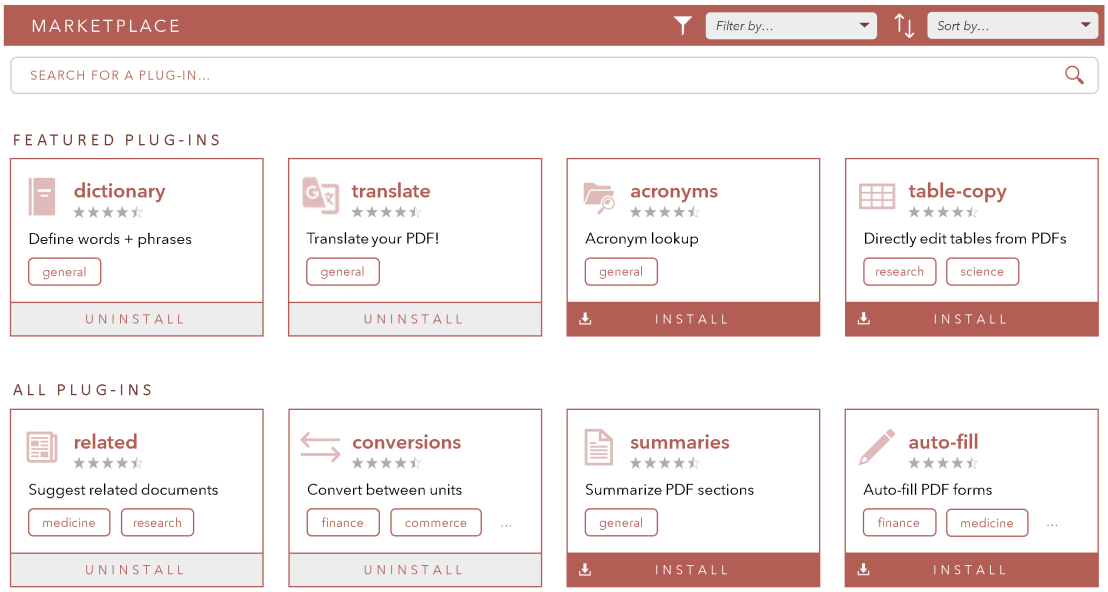


**Figure 10:** UI Prototype of our proposed plug-in marketplace for the next-gen document reader. Users can discover new plug-ins via the search bar or featured section, filter or sort the results, and (un)install plug-ins at any time.

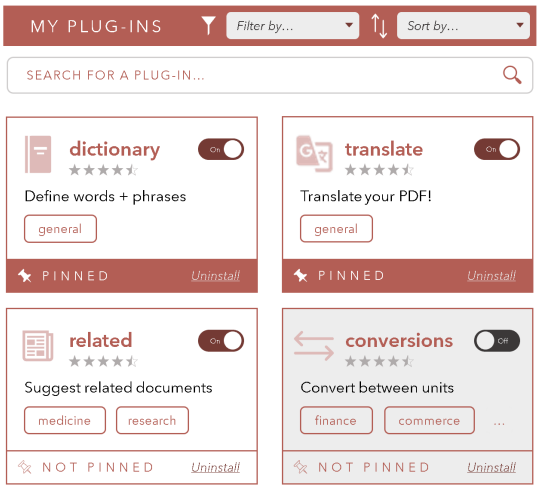


**Figure 11:** "My Plugins" page inside plug-in marketplace

consideration include summarization [20, 21], section title generation [22, 23], key sentence highlighting [24, 25], and question-answering [26, 27]. These ideas are more challenging to realize at the moment and may not be mature enough to be released to the general public, but recent progresses with large language models are making some of these features more achievable [28, 29].

## 5. Conclusion

In this paper, we present our **vision for the next-gen document reader** that will transform static, isolated documents into connected, trustworthy, interactive sources of information. This vision includes 12 **open-domain** and 6 **domain-specific** features powered by NLP, which can be accessed by the user through contextual plug-in pop-up menus while reading digital files. To allow users to customize their reading experiences with document readers, we also propose a centralized **plug-in marketplace** inspired by modern IDEs and text editors. Next steps include conducting formal user studies to further hone our UI prototypes (github.com/catherinesyeh/nextgen-prototypes) and vision, while also considering additional complex features to improve user-document interaction such as filtering or question-answering. We hope this work inspires and excites others about the future of document readers.

## References

[1] A. Head, K. Lo, D. Kang, R. Fok, S. Skjonsberg, D. S. Weld, M. A. Hearst, Augmenting scientific papers with just-in-time, position-sensitive definitions of terms and symbols, in: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, 2021, pp. 1–18.

[2] T. August, L. L. Wang, J. Bragg, M. A. Hearst,

A. Head, K. Lo, Paper plain: Making medical research papers approachable to healthcare consumers with natural language processing, arXiv preprint arXiv:2203.00130 (2022).

[3] R. Fok, A. Head, J. Bragg, K. Lo, M. A. Hearst, D. S. Weld, Scim: Intelligent faceted highlights for interactive, multi-pass skimming of scientific papers, arXiv preprint arXiv:2205.04561 (2022).

[4] A. Mostafavi, Sioyek, https://github.com/ahrm/sioyek, 2021.

[5] A. Veyseh, F. Dernoncourt, D. Dou, T. Nguyen, A joint model for definition extraction with syntactic connection and semantic consistency, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, 2020, pp. 9098–9105.

[6] S. Spala, N. Miller, F. Dernoncourt, C. Dockhorn, SemEval-2020 task 6: Definition extraction from free text with the DEFT corpus, in: Proceedings of the Fourteenth Workshop on Semantic Evaluation, International Committee for Computational Linguistics, Barcelona (online), 2020, pp. 336–345. URL: https://aclanthology.org/2020.semeval-1.41. doi:10.18653/v1/2020.semeval-1.41.

[7] A. P. B. Veyseh, F. Dernoncourt, T. H. Nguyen, W. Chang, L. A. Celi, Acronym identification and disambiguation shared tasks for scientific document understanding, arXiv preprint arXiv:2012.11760 (2020).

[8] A. Pouran Ben Veyseh, F. Dernoncourt, Q. H. Tran, T. H. Nguyen, What does this acronym mean? introducing a new dataset for acronym identification and disambiguation, in: Proceedings of the 28th International Conference on Computational Linguistics, International Committee on Computational Linguistics, Barcelona, Spain (Online), 2020, pp. 3285–3301. URL: https://aclanthology.org/2020.coling-main.292. doi:10.18653/v1/2020.coling-main.292.

[9] A. P. B. Veyseh, N. Meister, F. Dernoncourt, T. H. Nguyen, Acronym extraction and acronym disambiguation shared tasks at the scientific document understanding workshop (2022).

[10] A. P. B. Veyseh, F. Dernoncourt, W. Chang, T. H. Nguyen, MadDog: A web-based system for acronym identification and disambiguation, in: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations, Association for Computational Linguistics, 2021, pp. 160–167. URL: https://aclanthology.org/2021.eacl-demos.20. doi:10.18653/v1/2021.eacl-demos.20.

[11] V. Lai, A. Pouran Ben Veyseh, F. Dernoncourt, T. Nguyen, SemEval 2022 task 12: Symlink - linking mathematical symbols to their descriptions, in: Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022), Association for Computational Linguistics, Seattle, United States, 2022, pp. 1671–1678. URL: https://aclanthology.org/2022.semeval-1.230. doi:10.18653/v1/2022.semeval-1.230.

[12] V. D. Lai, A. P. B. Veyseh, F. Dernoncourt, T. H. Nguyen, Symlink: A new dataset for scientific symbol-description linking, arXiv preprint arXiv:2204.12070 (2022).

[13] Y. Deng, A. Kanervisto, J. Ling, A. M. Rush, Image-to-markup generation with coarse-to-fine attention, in: International Conference on Machine Learning, PMLR, 2017, pp. 980–989.

[14] J.-W. Wu, F. Yin, Y.-M. Zhang, X.-Y. Zhang, C.-L. Liu, Image-to-markup generation via paired adversarial learning, in: M. Berlingerio, F. Bonchi, T. Gärtner, N. Hurley, G. Ifrim (Eds.), Machine Learning and Knowledge Discovery in Databases, Springer International Publishing, Cham, 2019, pp. 18–34.

[15] J. Zhang, J. Du, Y. Yang, Y.-Z. Song, S. Wei, L. Dai, A tree-structured decoder for image-to-markup generation, in: International Conference on Machine Learning, PMLR, 2020, pp. 11076–11085.

[16] T. Ito, K. Tsubouchi, H. Sakaji, T. Yamashita, K. Izumi, Contextual sentiment neural network for document sentiment analysis, Data Science and Engineering 5 (2020) 180–192.

[17] M. Rhanoui, M. Mikram, S. Yousfi, S. Barzali, A cnn-bilstm model for document-level sentiment analysis, Machine Learning and Knowledge Extraction 1 (2019) 832–847.

[18] J. A. Teixeira da Silva, H. Bornemann-Cimenti, Why do some retracted papers continue to be cited?, Scientometrics 110 (2017) 365–370.

[19] M. J. Bolland, A. Grey, A. Avenell, Citation of retracted publications: A challenging problem, Accountability in Research 29 (2022) 18–25.

[20] A. Cohan, F. Dernoncourt, D. S. Kim, T. Bui, S. Kim, W. Chang, N. Goharian, A discourse-aware attention model for abstractive summarization of long documents, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), Association for Computational Linguistics, New Orleans, Louisiana, 2018, pp. 615–621. URL: https://aclanthology.org/N18-2097. doi:10.18653/v1/N18-2097.

[21] H. Lee, C. Park, S. Yoon, T. Bui, F. Dernoncourt, J. Kim, K. Jung, Factual error correction for abstractive summaries using entity retrieval, arXiv preprint arXiv:2204.08263 (2022).

[22] N. Rizvi, S. Gehrmann, L. Wang, F. Dernoncourt, Margin call: an accessible web-based text viewer with generated paragraph summaries in the mar-

gin, in: Proceedings of the 12th International Conference on Natural Language Generation, Association for Computational Linguistics, Tokyo, Japan, 2019, pp. 244–246. URL: https://aclanthology.org/W19-8632. doi:10.18653/v1/W19-8632.

[23] S. Gehrmann, S. Layne, F. Dernoncourt, Improving human text comprehension through semi-Markov CRF-based neural section title generation, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 1677–1688. URL: https://aclanthology.org/N19-1168. doi:10.18653/v1/N19-1168.

[24] S. Spala, F. Dernoncourt, W. Chang, C. Dockhorn, A comparison study of human evaluated automated highlighting systems, in: Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation, Association for Computational Linguistics, Hong Kong, 2018. URL: https://aclanthology.org/Y18-1071.

[25] S. Spala, F. Dernoncourt, W. Chang, C. Dockhorn, A web-based framework for collecting and assessing highlighted sentences in a document, in: Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations, Association for Computational Linguistics, Santa Fe, New Mexico, 2018, pp. 78–81. URL: https://aclanthology.org/C18-2017.

[26] Q. H. Tran, N. Dam, T. Lai, F. Dernoncourt, T. Le, N. Le, D. Phung, Explain by evidence: An explainable memory-based neural network for question answering, in: Proceedings of the 28th International Conference on Computational Linguistics, International Committee on Computational Linguistics, Barcelona, Spain (Online), 2020, pp. 5205–5210. URL: https://aclanthology.org/2020.coling-main.456. doi:10.18653/v1/2020.coling-main.456.

[27] J. Xiao, L. Wang, F. Dernoncourt, T. Bui, T. Sun, J. Han, Open-domain question answering with pre-constructed question spaces, in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop, Association for Computational Linguistics, Online, 2021, pp. 61–67. URL: https://aclanthology.org/2021.naacl-srw.9. doi:10.18653/v1/2021.naacl-srw.9.

[28] T. Goyal, J. J. Li, G. Durrett, News summarization and evaluation in the era of gpt-3, arXiv preprint arXiv:2209.12356 (2022).

[29] T. Zhang, F. Ladhak, E. Durmus, P. Liang, K. McKeown, T. B. Hashimoto, Benchmarking large language models for news summarization, arXiv preprint arXiv:2301.13848 (2023).