SUPPORTING COLLABORATIVE AUTHORING AND REVIEWING WORKFLOWS IN DESKTOP-BASED AND WEB-BASED AUTHORING ENVIRONMENTS

By

MAHDI JABERZADEH ANSARI

B.E., University of Isfahan, 2008

A THESIS

Submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Enterprise Information Systems Department
Institute for Informatics

UNIVERSITY OF BONN
Bonn, Germany

2016

Approved by:

Prof. Dr. Rainer Manthey
Dr. Christoph Lange

Copyright

Abstract

Research and producing knowledge is what researchers and scientists are occupied with. However, even if someone achieves the best outcomes in a research and he or she doesn't publish the results, it is like that he or she did nothing. Thus, effective research is meaningful when the findings are communicated with other scientists in a formal way which is called *scientific writing*. "Scientific writing is the art of presenting research ideas clearly, documenting results precisely, and drawing implications correctly." (Peat, Elliott, Baur, & Keena, 2002, p. 254).

For many years type machines were the only classical way of producing a typed copy of a document. Furthermore, writing with a pen on the margins of the papers was the most popular way of commenting on a piece of document. Later by appearing the computers, type machines had been substituted by computers with an installed word processor software. However, it took some years, which reviewing tools appear in some software tools. In this way, the digital world changed the authoring world. Nowadays most of the authors try to type directly in digital forms instead of writing by stationery and then typing the handwritten documents by typists. Moreover, gradually collaborative spirit changed the expectations from authoring tools and some versioning software tools tried to bring this ability inside of the authoring tools. Assume Alice and Bob are trying to form a document. For some reason, each of them is responsible for writing some parts of that document. However, both of them are responsible for the correctness of the whole document. Thus, in a traditional way each of them types his or her parts and finally merge two files by copy and paste into one file and then review whole together. This scenario can happen in a scientific writing as well. Collaborative writing and peer reviewing are two major activities which happen in the process of creation of many scientific papers. Thus, the need for such a comprehensive authoring tool with collaborative and peer reviewing abilities is the focus of developers for improving authoring tools.

Nowadays, by the presence of the internet in almost all the scientific institutions, collaboration got a new form, as well as scientists with thousands of miles distance, can work together. Furthermore, as web-based tools are OS independent, many web-based tools became popular among the users. Therefore, the development has trended towards the development of the web-based products during the past decade. However, one of the advantages of the desktop-based tools which survives them yet in the market is that they can be used everywhere on portable devices (e.g. laptop or tablet) even without connecting to the internet. Thus, we are faced with two types of authoring environments and two types of the end users in connection

to authoring tools. Some users prefer web-based authoring software tools and some others prefer desktop-based ones, however, both of them have a common requirement which is the collaborative ability in authoring tools. This collaboration is not only in the form of writing together, but also reviewing of other's writings is a form of collaboration as well.

There is a demand among the users of web-based and desktop-based authoring and reviewing environments to collaborate on writing and reviewing scientific papers in their preferred software tools. Thus, it is required to make these two types of software tools as closer as possible to each other. This thesis was aimed to help developers to provide better facilities for scientific writing in their authoring and reviewing software products, detect the shortages and limitations in both types of environments, and finally provide a better understanding of the pros and cons of different technologies that are used in both environments.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would first like to thank all of my teachers who conducted me in the way of education. From the first year of the elementary school in 1992 till today. To all those people who encouraged me during these years.

A warm thank to Germany, its nation, and all the professors at the University of Bonn who gave me this chance to study here. It was not just three years of study, it was a chance for learning about how to be a positive human, how to love and how to live in a way that being more positive for the society.

I would like to acknowledge my supervisors Prof. Dr. Rainer Manthey and Dr. Christoph Lange for their valuable supports. The door to Dr. Lange office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this thesis to be my own work, however, steered me in the correct direction whenever he thought I needed it. Furthermore, warm thanks to my mentor Mr. Afshin Sadeghi one of the Ph.D. students in the EIS Group of the University of Bonn, who helped me a lot during the writing of this master thesis. I am gratefully indebted to his for his valuable comments on this thesis.

I have to thank the Honestly Company and especially Mr. Sebastian Wentzel that supported our online survey by providing a free access to their feedback collector system. As well, to all those expert people who participated in our survey and supported us during this process.

And last but not least, thanks to Mr. Johannes Wilm the main programmer of the Fidus Writer project because of his technical supports during our working with Fidus Writer's file format and all the other people who are part of the OSCOSS project. I would also like to thank the experts who were involved in the validation survey for this research project from the different parts of the world. Without their passionate participation and input, the validation survey could not have been successfully conducted.

Mahdi Jaberzadeh Ansari

Dedication

This thesis is dedicated to all those people who like knowledge. To those people who like to study, however, they cannot reach what they are deserved to. To my parent, especially to my mother who was the most important sympathetic of my study path. And finally to my spouse, Tala, for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Chapter 1 - Introduction

The term collaborative writing refers to a group act of two or more specialist authors which work together to form some pieces of scientific written. Collaborative writing usually refers to the writing of the scientific and academic papers, industry and government writings (Lowry, Curtis, & Lowry, 2004). If people gather together and write a story or roman collaboratively, that activity is usually called collaborative fiction (Marriage of Minds: Collaborative Fiction Writing, 2000). Both of these two collaborative activities can use same authoring tools. However, as the process of forming the final article is different, thus, their requirements are different as well. Therefore, in this thesis, we focus on supporting collaborative writing for scientific purposes.

Collaborative writing has some benefits, such as learning, increasing the quality by decreasing the faults, and much more which are not part of the objects of this thesis.

Scientific collaborative writing usually is accompanied by other activities which are called scholarly peer reviewing and collaborative document reviewing. Scholarly peer reviewing is the activity of evaluation of an academic paper by someone of similar competence to the subject of the paper (Spier, 2002). Scholarly peer reviewing is a collaborative activity by itself as the author(s) of the paper will receive the result of the evaluations and must edit the paper based on the suggestions if acceptable. During the process of accepting or rejecting one suggestion, it may happen that the author(s) and reviewer require discussing their own reasons or refer to pieces of evidence. This collaboration can be even broader if the number of the reviewers is more than one reviewer. In this case, the reviewers can receive each other's notes, corrections or suggestions and they can reject or approve them.

Therefore, it is clear that why there is a considerable demand for collaborative authoring and reviewing tools, especially among scientific societies. Furthermore, it is easier if they access to some tools tailored for writing scientific papers collaboratively.

Collaborative authoring and reviewing desktop-based tools are not such new products in the digital world. In fact, the first instance of a collaborative real-time desktop-based editor for personal computers was "Instant Update" which was released in 1991 for Mac OS by ON Technology, later a Microsoft Windows version of the product published as well (Instant Update, 2016).

During the last decade by appearing the web 2.0 phenomenon, a huge demand for web-based document editing tools was made. Slowly not only online editing tools became popular, but also adding the collaborative ability of editing documents made them more popular as well.

Before appearing the collaborative web-based authoring tools, it was desktop-based authoring tools which were broadly used by the end users. At that point of time, most of the authoring software tools were classified to WYSIWYG[1]like Microsoft Word and non-WYSIWYG like LaTeX. Most of the instances of these two types of products, and particularly MS Word and LaTeX are single user authoring tools. Thus, most of the software companies which produced authoring tools, more focused on different editing and styling features rather than collaborative ability.

In the interim version control software appeared. For example, SVN[2] or Git (A Short History of Git, 2016)[3]. Version control software tools provide facilities for managing of the changes in simple text files in a collaborative environment. Using LaTeX in the combination of versioning tools helped to bring the sprite of collaboration to non-collaborative authoring software tools.

Git was designed to provide a management mechanism on changes of programming source files, which are usually simple text files. In fact, Git keeps a copy of the file in a repository in a safe situation. Each user can clone the file in his or her local machine, make the changes, commit the changes to the local copy and finally push the changed file back to the repository as a newer version of that file. If two or more users change same part of the same file simultaneously, in the time of pushing, the latest user who wants to push his or her changes to the repository will receive a conflict error and has to resolve the conflict by editing the file and push it again. Here is an example to clarify this process. Assume Bob and Alice modified same part of the file "Redme.txt". Both of them started working on the latest version of that file. Bob has finished his changes faster and he can commit and push the changes to the repository without any error (Part B of Figure 1-1).

When Alice wants to push her changes in the repository, she will receive a conflict error as it is illustrated in part D of Figure 1-1. It illustrates the current condition of the second line of the file in the latest version in the repository (HEAD) and what the changes that Alice made are (after equal signs). She has to decide and select one of the changes, maybe she selects her changes, or Bob's changes, or a combination of both. Finally, she has to remove the three lines that were added by Git, and mark the file as resolved and push it again.

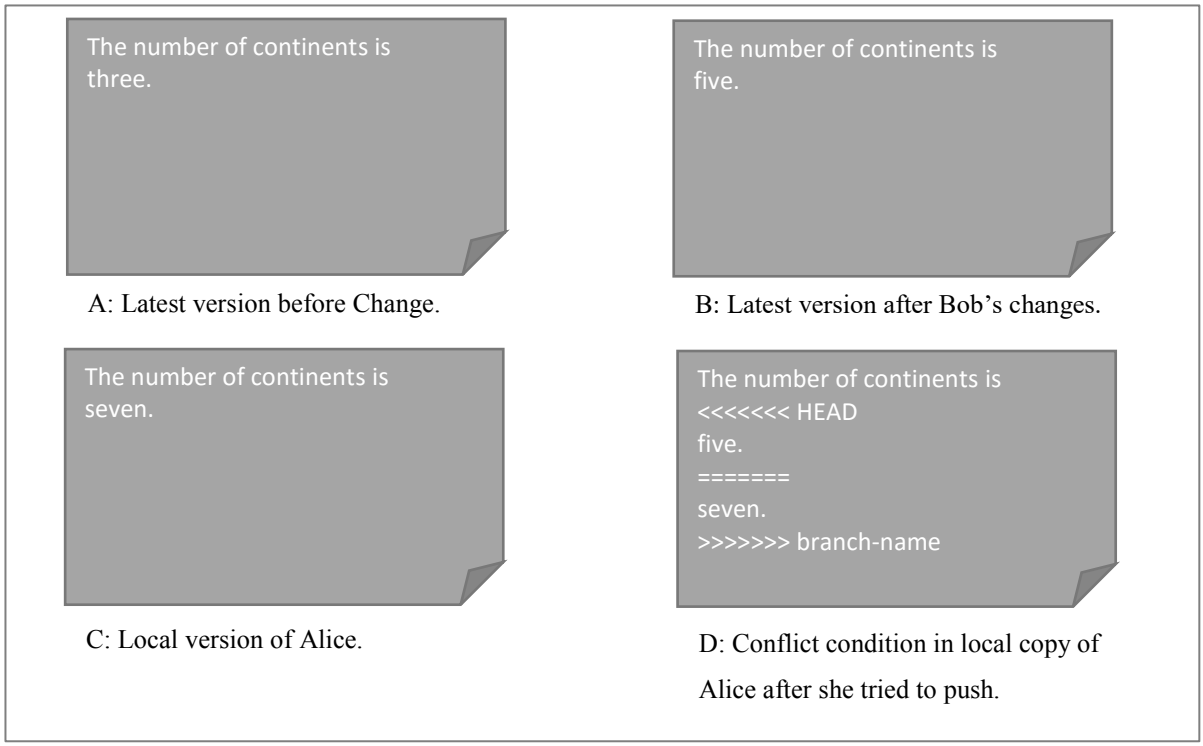
---

[1] What You See Is What You Get (Definition of WYSIWYG in English, 2016).

[2] More information about SVN can be found here: https://subversion.apache.org/

[3] More information about Git and its history can be found here: https://git-scm.com/

Figure 1-1: A sample workflow in changing a file in Git



This process works fine for the programming source code files and any other type of files which are plain text. Thus, using LaTeX in the combination of Git or any other version control tools could be satisfied the intuitive requirements for collaboration in editing the text files. However, even using version control tools with LaTeX has its own limitations. For example, in source code files the unit is a line of code and Git algorithm was designed to compare lines of two text files and illustrate the changes between two lines of code, while when the files are documents usually some concepts like paragraph are being to exist. Furthermore, it may happen that a user only changes some words of a line or some parts of a paragraph and usually when other users want to know who change what, it is important that the versioning software marks exactly those changed words and not the whole line or the whole paragraph. Thus, we can see this solution is not a perfect solution for collaboration on editing document files. This story can be worse when the files are some binary files like Microsoft Word files (i.e. DOCX files) and Git is not an acceptable solution for binary files.

Therefore, there was a demand for authoring tools that provide version control and collaborative ability by their own selves. This coincided with the appearing of web-based authoring tools like Google Docs. In the beginning and even now while many web-based authoring tools have provided strong version control and collaborative features, they have suffered from the limitations in their editing and styling options. For example, one of the

unresolved issues in all web-based authoring tools is supporting multicolumn page layout. In fact, capabilities of web-based authoring tools are limited by the browsers abilities and HTML features and other browser-based programming languages like JavaScript. Thus, even now there are many people who prefer to use desktop-based authoring tools and if they require collaborating with other people, they will share the document files via file sharing systems like Dropbox. Thus, the needs for supporting collaborative ability and benefits from the powerful features of desktop-based authoring tools at the same time, caused software companies to add collaborative ability to desktop-based authoring tools. Particularly MS Word, which is the most popular authoring tools added the real-time collaborative ability from the version 2016 (LOPEZ, 2015).

One of the aims of this thesis is discussing the expectation of the scientific communities from authoring and reviewing tools. To achieve this goal, we organized an online survey and invited scientists to participate in. The result of that survey will be discussed in Chapter 2.

Furthermore, we investigated the contents of the 27 scientific papers and listed used features of them. By doing a statistical analysis on the result of the investigation, we prioritized the features that an authoring tool has to support to be useful for scientific authoring. Moreover, we investigated a subset of web-based and desktop-based authoring tools. We will discuss in detail about different supported features of those authoring tools and with the help of the result of our online survey, we looked for the reasons of popularity and unpopularity of some. The result of these two investigations reflected in Chapter 3.

Another problem is how to make a connection between web-based and desktop-based authoring tools. As we will see in Chapter 3, some people prefer to use web-based authoring tools and some other prefer to use desktop-based tools. Thus, the collaboration between them is difficult. It is not almost all time to convince someone to use online MS Word or desktop version of it, sometimes someone is more comfortable to use a product from one company and someone else prefers to use a product of another company. The easiest and fastest way of making a bridge between two products of two different company is to make a converter that converts files between these two software tools. Making a converter between two different types of files has its own challenges. It can be seen that Microsoft Word is the most dominant authoring tools, and almost all of the popular authoring tools support import and export of DOCX files. Thus, we decided to investigate the methods for converting to and from “DOCX” file type to other types. On the other hand, we selected “FIDUS” file type as it is the standard file type of Fidus Writer which is one of the newest and under development web-based tools for academic writing in a collaborative environment. FIDUS files are the compressed files of

some “JSON” files. JSON is a structure that is supported strongly by JavaScript and almost all the major web browsers. This property makes JSON a suitable type of files for web-based tools. We developed a Java based application for converting files from the FIDUS type to the DOCX type and vice versa. The idea is that one user forms a document file in Fidus Writer, and requires one of his colleagues editing the file or make some comments on it. Thus the question is how can we do this conversion with the less data losing? Furthermore, based on our investigation which has been illustrated in Table 3-1, 70% of people who use Microsoft Word don’t know how to make some essential features like “Title” inside of a DOCX document. So another challenge was that how to detect elements inside of the DOCX files and import those in FIDUS file in a correct way. We evaluated different libraries and methods that are exist for working with DOCX files and will discuss what are the pros and cons of “docx4j” library that we selected for our works. Furthermore, in this section, we will compare different methods that web-based authoring tools use for saving documents files.

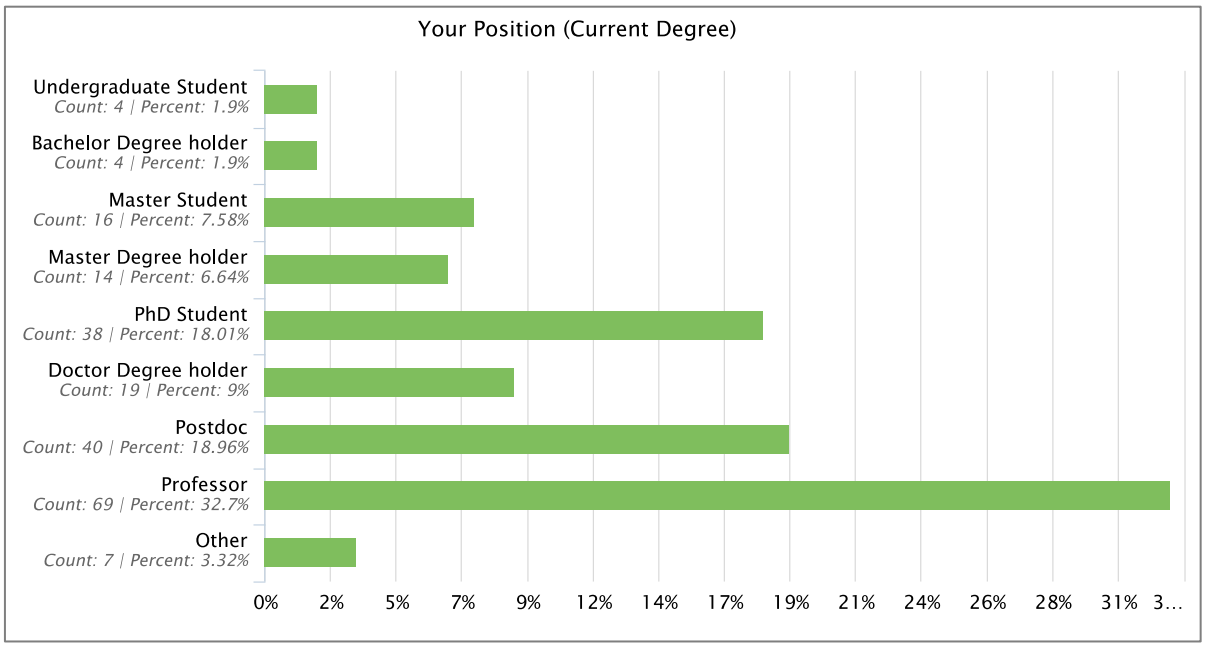
Finally, in Chapter 6 we will discuss that, what the essential features are for an authoring and reviewing tool to be used for writing an academic document, what the challenges are in importing from and exporting to other file types for a document, and finally what the key features are in popularity or unpopularity of an authoring tool, particularly between scientific communities.

Chapter 2 - Requirement Assessment

Experiences and Expectations of Scientific Authors

For supporting web-based and desktop-based authoring tools and to provide better facilities for scientific authoring, the best way is, using the experiences of those end users who use these kinds of tools for scientific writings. Thus we organized an online survey[4] and invited academic staff of different universities via email. 213 people participated in our survey[5]. Fortunately, almost 80% of the participants were Ph.D. students, doctor degree holder or above. Figure 2-1 illustrates the distribution of participated people based on their academic degrees. People participated from more than 60 universities in 15 different countries. More details can be found in Appendix A.

Figure 2-1: Educational level of the participants in our online survey.



The first questions that we were looking for their answers, were how many of the interviewees had experienced collaborative scientific writing, and of course, how many of them had experienced peer reviewing. Thus we asked the following two questions:

---

[4] We used the Honestly.de Company's web-based software. They produce some product to collect feedback for their clients and analysis collected data. They provided a free access to their product for us because of helping of this scientific research. So almost all of the charts of Chapter 2 were made by their analyzer tool.

[5] The result of the online survey is available in EIS repository with this address: Data/online_survey_export.xlsx

- *Have you ever participated in a collaboration to produce a piece of scientific writing?*
- *Have you ever been involved in a peer review?*

The result of these two questions are illustrated in Figure 2-2 and Figure 2-3 respectively.

Figure 2-2: Distribution of the participants who had experienced collaborative scientific writing

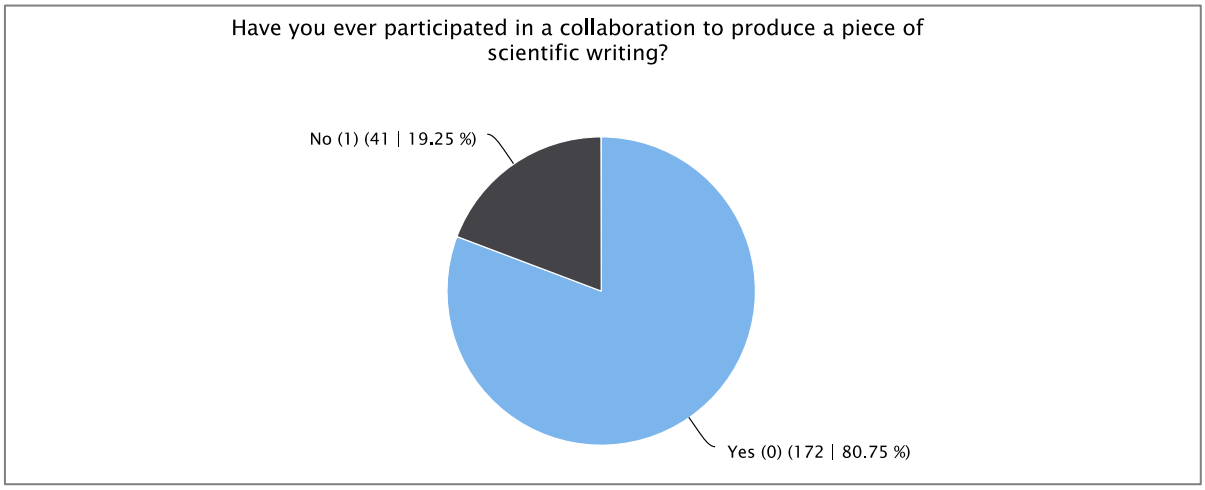
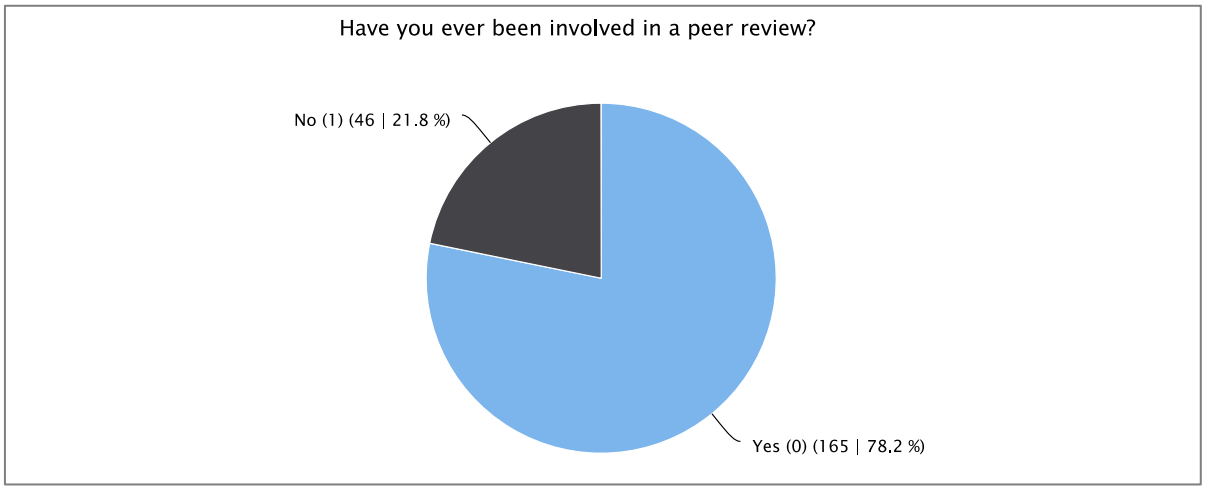


Figure 2-3: Distribution of the participants who had experienced peer reviewing



As it can be seen in Figure 2-2 and Figure 2-3 almost 80% of participants had experienced collaborative writing and peer reviewing. Knowing the percentages of the expert people in these two areas was important for us because these numbers demonstrate how much the result of the survey is trustable. When the interviewees have enough experience related to the subject of a survey, that survey is trustable, otherwise, the selected statistical society is not fit enough to wind up based on the result of that survey.

The second reality that we were looking for was that which authoring software tools are more popular among scientists for creating scientific writings. Thus we asked the following two questions.

- *Which of the following desktop-based/web-based authoring tools have you used for scientific purposes?*

The results of these two questions have been demonstrated in Figure 2-4 and Figure 2-5. Please note that participants could select more than one tools. To obtain a better understanding of popularity and widely use of desktop-based and web-based tools in comparison to each other, a list of all authoring tools in both category and their votes has been collected in Table 2-1.

Table 2-1: Number of the votes for different authoring tools

Category	Votes	Type	Category	Votes	Type
Microsoft Word	182	Desktop	**Mathematica Online**	4	Web
Google Docs	106	Web	**Polaris Office**	4	Desktop
TeX/LaTeX	93	Desktop	**GNU TeXmacs**	3	Desktop
Microsoft Office 365	42	Web	**Scrivener**	3	Desktop
Microsoft Office Word Online	34	Web	**ThinkFree (FreeOffice) Write**	3	Desktop
Other (Desktop & Web-based)	52	Desktop, Web	**Zoho Docs**	3	Web
LibreOffice Writer	29	Desktop	**AbiWord**	2	Desktop
Overleaf	20	Web	**Fidus Writer**	2	Web
ShareLaTeX	18	Web	**Calligra Words**	1	Desktop
Apache OpenOffice Writer	15	Desktop	**Mellel**	1	Desktop
Mathematica	15	Desktop	**Nisus Writer**	1	Desktop
WordPerfect	14	Desktop	**Authorea**	1	Web
EtherPad	8	Web	**Gobby**	1	Web
LyX	8	Desktop	**IBM Connections Docs Cloud**	1	Web
WPS Office Writer	5	Desktop	**WriteOnline**	1	Web

Please note that there were some tools that had no votes and we removed them from Table 2-1.

It can be seen that Microsoft Word between desktop-based and Google Doc between web-based authoring tools are the most popular authoring tools. Furthermore, it can be seen that desktop-based authoring tools are more popular than web-based ones. We will discuss in Chapter 3 each of these authoring tools in more details. We will compare their capabilities and look for the reasons that why some authoring tools are more popular among scientists for producing scientific writings.

Figure 2-4: Amount of the using different desktop-based authoring tools for scientific writing

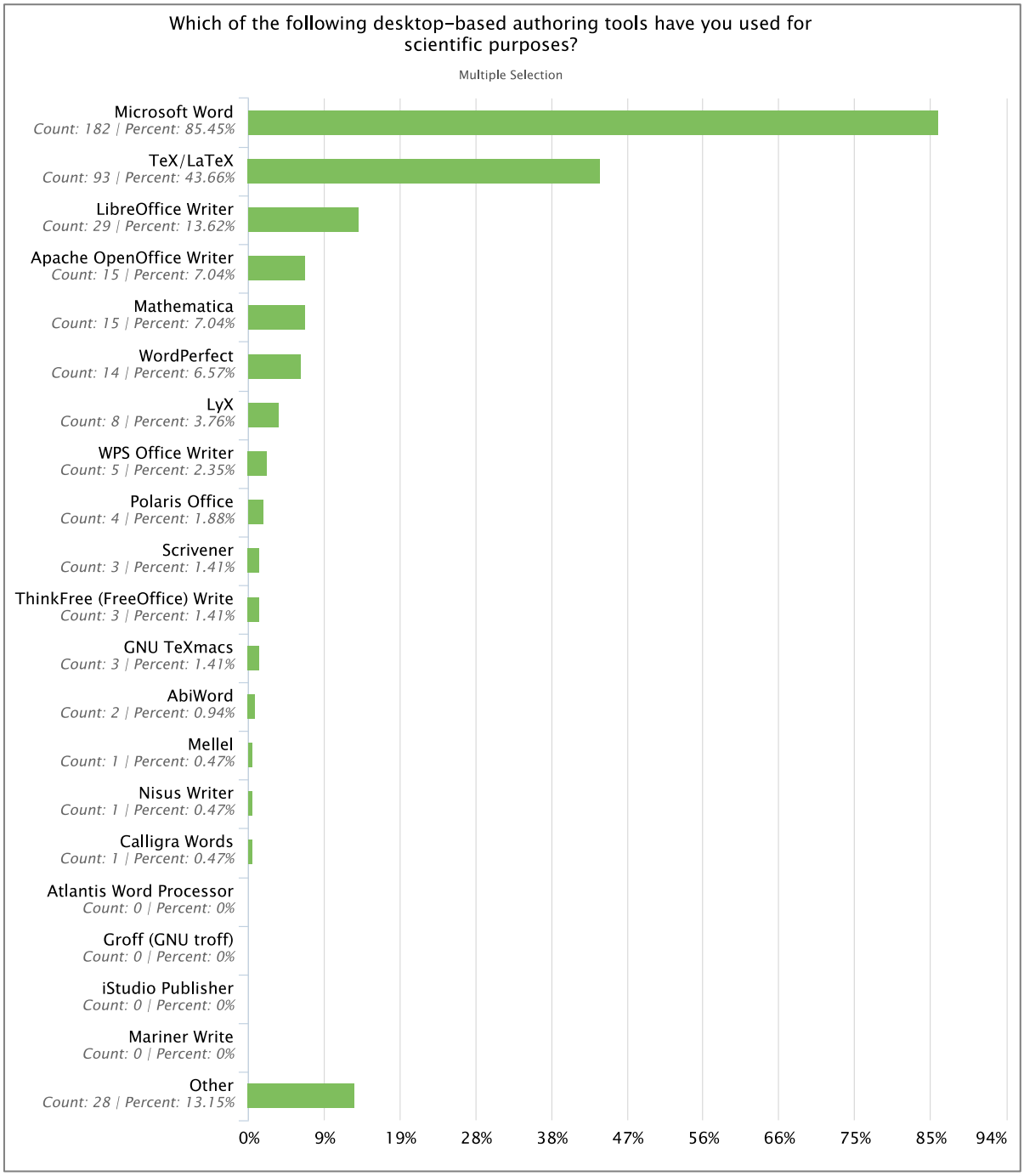
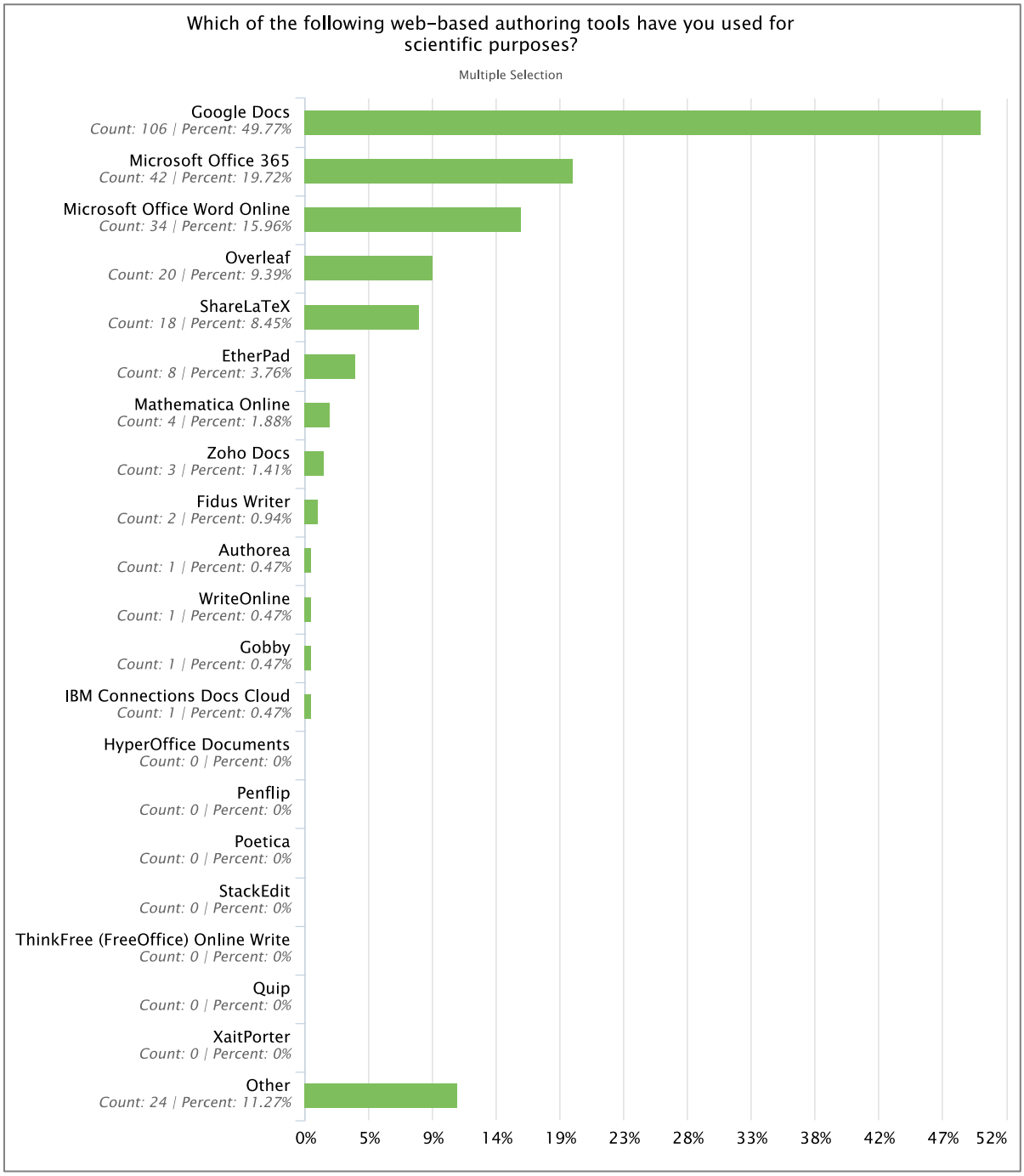


Figure 2-5: Amount of the using different web-based authoring tools for scientific purposes



We asked the interviewees to give a rating between 0 to 5 to each of the tools that they used and tell us to what extent those tools were suited to their requirements, or to what extent they were satisfied with those tools. They could select between 0 to 5 stars for each tool. The result of this question reflected in a chart in Figure 2-7. The result sorted based on the number of the total votes. One reality that can be concluded from this chart is that in average people are more satisfied from Microsoft Word rather than Google Docs. Furthermore, even between

different version of Microsoft word (i.e. desktop-based version or web-based ones) people are more preferred desktop-base versions rather than web-based ones while web-based version provided better supporting for collaboration, however, less supporting for editing features. Another reality is that WYSIWYG authoring tools are more popular rather than some powerful tools like LaTeX. Thus, we can conclude the majority of the end users prefer powerful authoring tools with a WYSIWYG design.

The number of the people who used Authorea is definitely lower than other authoring tools. It has been claimed that Authorea was designed to become a "Google Docs for Academics" (Lomas, 2014). We expected at least a significant percentages of the scientists have used the Authorea at least for one time. However, based on our observation, people expect that a WYSIWYG editor is more or less similar to dominant WYSIWYG editors like MS Word. As much as a product gets away from our expectation its chance for success becomes lower and lower. The problem related to Authorea is its rich text box that the user must press a key to start a new paragraph, despite the fact that it has an effective facility for citation. Figure 2-6 illustrates the unusual behavior of the reach textbox of the Authorea. The user requires pressing 'Edit' or 'Insert blow' links to be able to add a new paragraph or edit an existing one. This extra clicks can make the users uncomfortable.

Figure 2-6: A view of the reach textbox of the Authorea

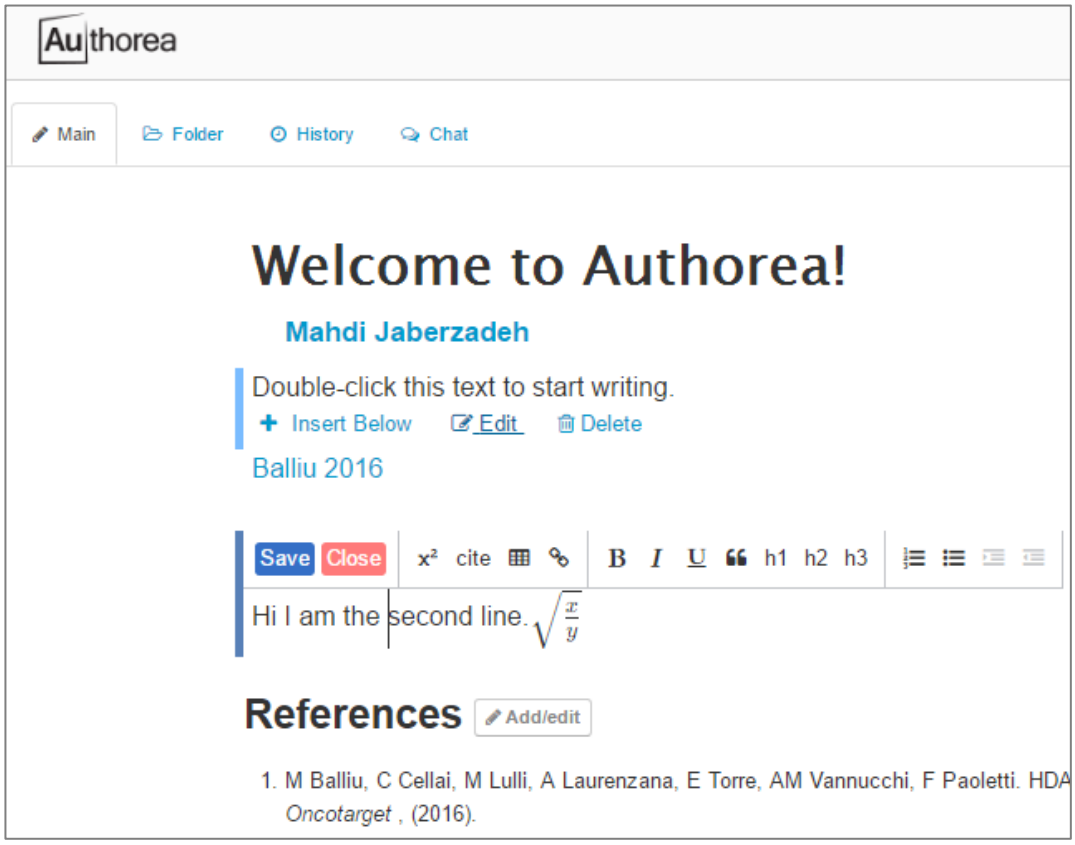


Figure 2-7 : Results of the satisfaction ratings on the different authoring tools

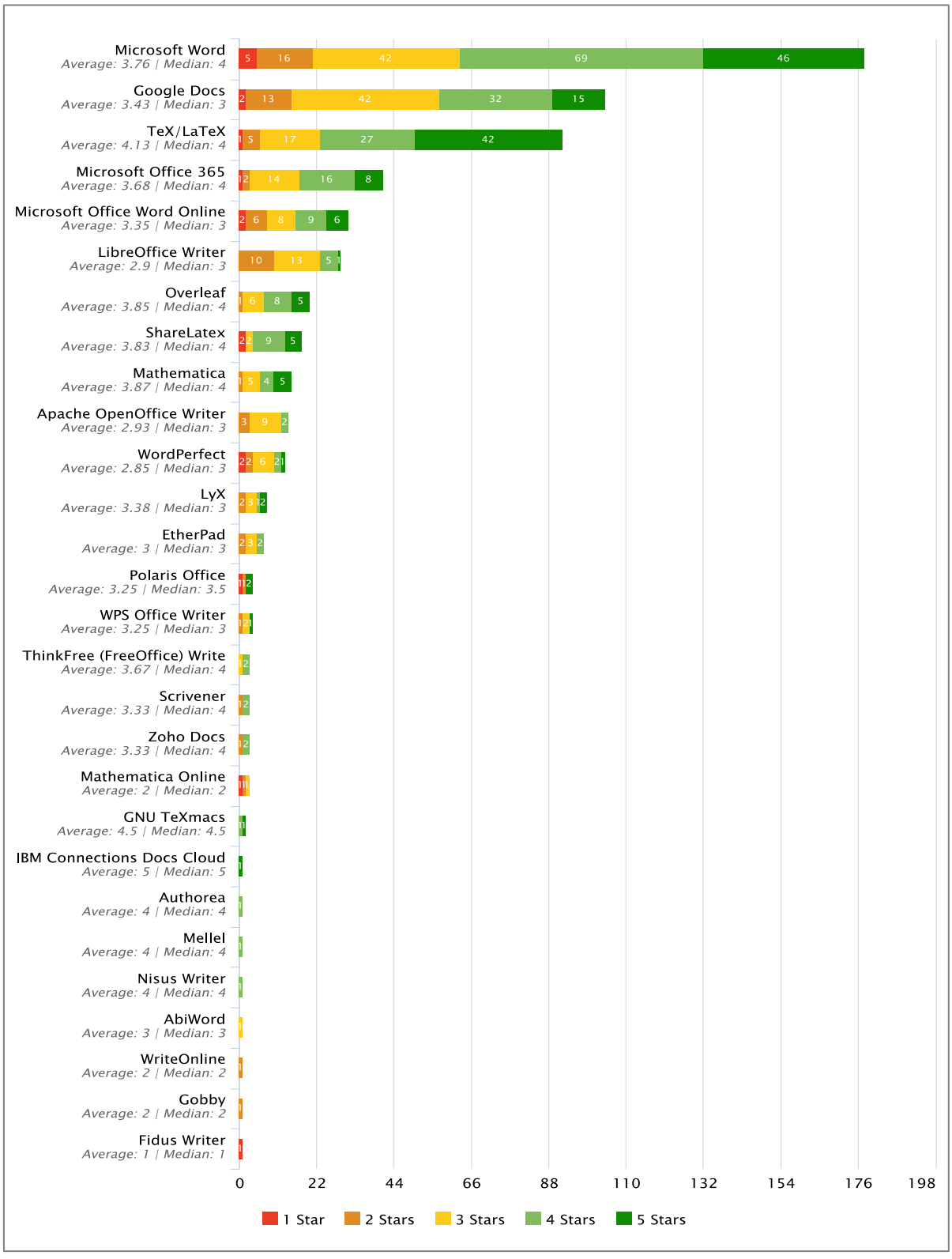
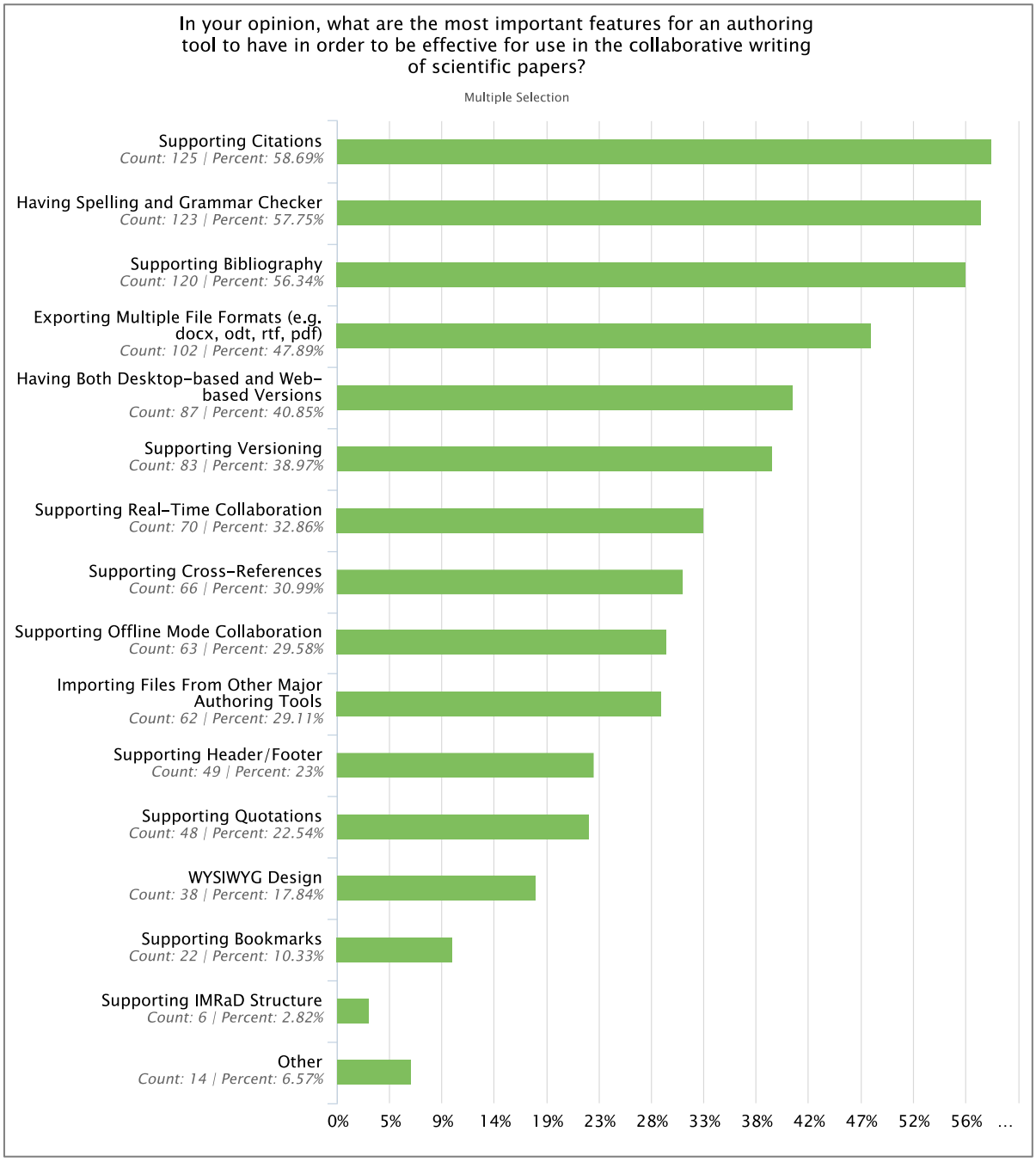


Figure 2-8: Opinion of the interviewees about the most important features of an authoring tool, which is suitable for scientific writings.

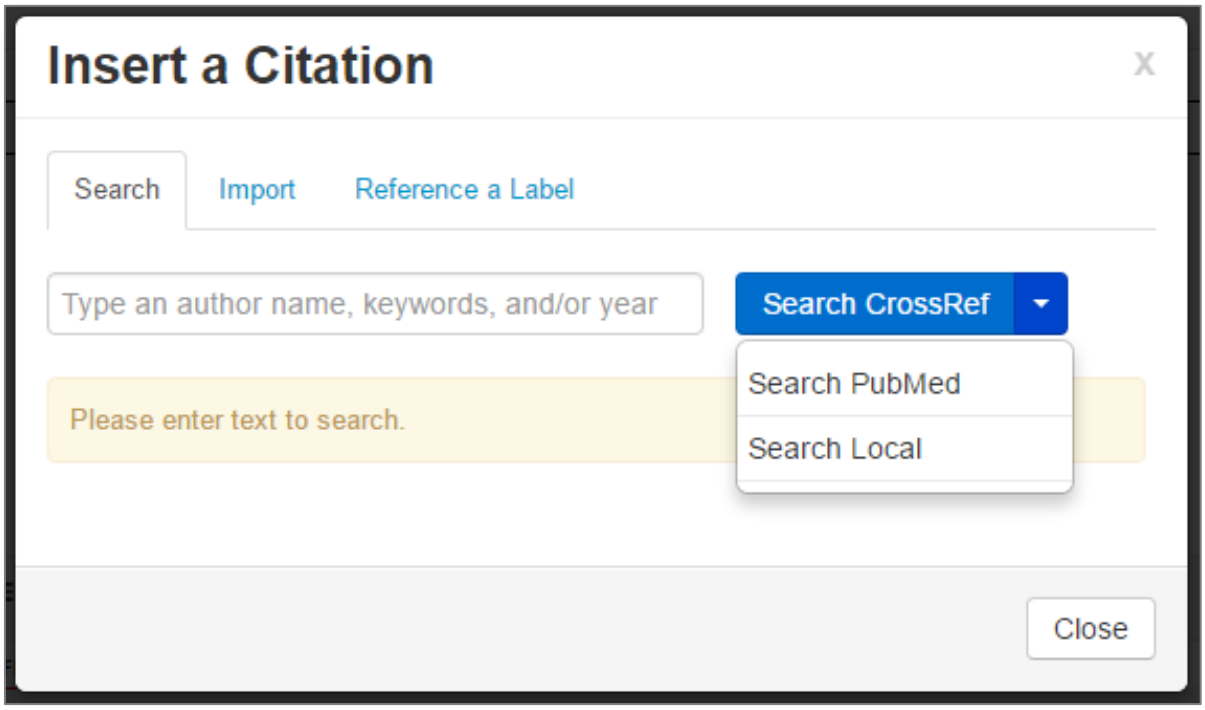


As we mentioned in Chapter 1, we investigated 27 real scientific papers and tried to know which features are more used in scientific writings. We will discuss the result of that investigation in Chapter 3. We designed a question based on the result of that investigation and asked the interviewees following question:

- *In your opinion, what are the most important features for an authoring tool to have in order to be effective for use in the collaborative writing of scientific papers?*

The result of this question is illustrated in Figure 2-8. Based on the result of our investigation on scientific papers, the citation is the most used feature in our sample documents. Moreover, we can see that most of our interviewees believed that supporting citation is the most important feature that a tool must have to be suitable for scientific writing. A positive example of what can be done for supporting citation with an efficient tool is the citation facility of Authorea. As it has been demonstrated in Figure 2-9, it is possible to type some part of the name of author(s), keywords or the year of publication, and it will provide a list of possible items, and with one click it imports to the document. This facility can help authors to prevent typing each property of a source in a form. There are some add-ons for Google Docs and MS Word, however, as we tested them, they are not efficient and in most of the cases cannot find any related item.

Figure 2-9: The citation tool inside of the Authorea



In fact, MS Word supports citation, however, the user must type at least five mandatory items, which is time-consuming and boring. So it was observed during our investigation on real

scientific papers that in most cases people preferred to insert citations as simple text from other citation tools like Google Scholar[6] or Mendeley[7]. Importing the citation as a simple text from a trustable source can be useful because mostly there will be no fault or typos in the cited text, however, maintaining of that citation will be difficult for further editing and it is required repeating this process again and again especially if a citation repeated in different parts of a document.

Therefore, it can be concluded that providing an efficient tool for citation with the ability of online search in any bibliography database can be helpful to make an authoring tool more popular among scientific communities.

Another interesting point of Figure 2-8 is that almost 50% of the interviewees expected the authoring tool could support exporting to and importing from other formats. For example, one of the interviewees mentioned:

> *"It would be wonderful if a shared authoring platform was able to integrate from multiple and export back to each."*

Another reality related to this chart is that supporting real-time collaboration is more interesting rather than the offline collaboration mode. One option that we forgot to ask and some of the interviewees mentioned that in "Other" option of the question is the ability to track the changes. When an authoring tool wants to provide real-time collaboration it is necessary to track the changes of the document by each user. Google Docs has a powerful facility for tracking the changes. A process for tracking the changes saves the new revision of the document every some seconds.

Another two questions that we asked from interviewees were more related to reviewing tools. First, we asked about their experiences and then about their expectations:

- *Which of the following methods or software tools would you prefer for peer reviewing?*
- *Which of the following features are most important for a specialized reviewing software tool?*

Figure 2-10 and Figure 2-11 illustrate the result of the above questions. It can be seen that PDF tools play a considerable role in reviewing documents. Major PDF tools like Adobe PDF Reader or Foxit Reader play a considerable role in reviewing by providing annotation and

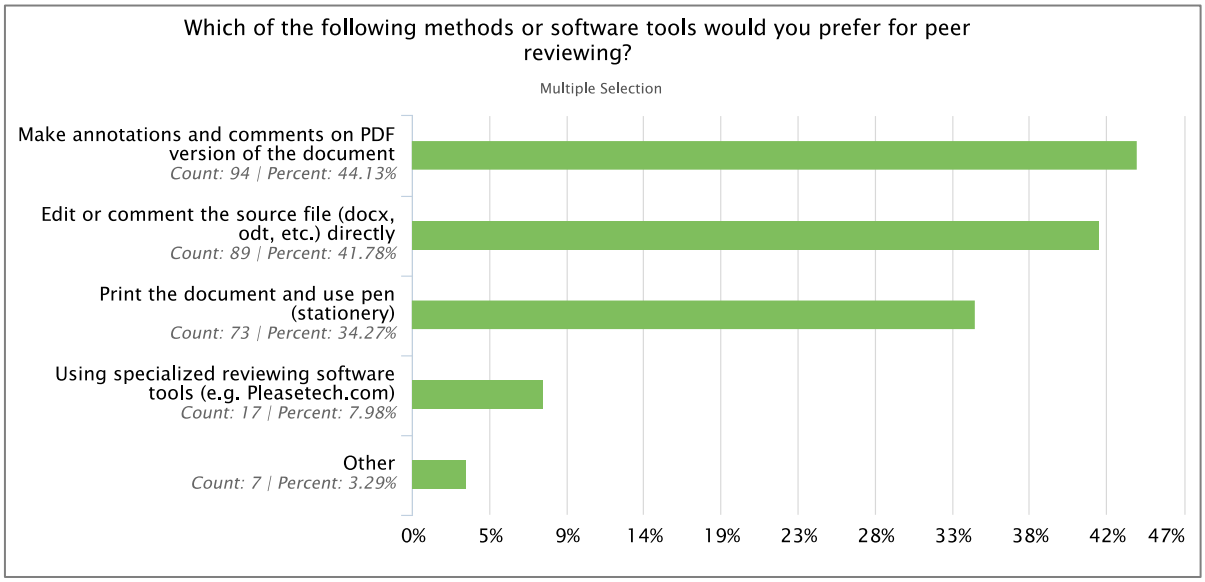
---

[6] Google Scholar is the biggest database of academic papers and books which is supported by Google. https://scholar.google.com/

[7] Mendeley is a free reference manager and academic social network. https://www.mendeley.com/

commenting tools. Another reality that can be extracted from Figure 2-10 is that almost 42% of the respondents known that some tools like MS Word have some reviewing options. However, even now 34% preferred to print the documents and use a pen for reviewing. It means that authors require spending much time for negotiating with reviewers and spend much more time to digitize the suggestion. Furthermore, selecting this method means we cannot track the changes and make a snapshot of how the evolution of the document.

Figure 2-10: Usual methods that the interviewees used for the peer reviewing



Finally, it can be seen just 8% were interested in using extra tools for reviewing. Thus, it can be concluded that people maybe prefer to use an integrated tool that supports authoring and reviewing both together.

Related to what our statistical society expected from a reviewing tool, Figure 2-11 illustrates supporting for commenting, highlighting and annotations are three main expectations. It can be seen supporting for offline reviewing is more interested rather than real-time ones while for authoring tools real-time collaboration was more interested. Furthermore, almost 75% thought providing different authority is not so much important. Again, here some interviewees suggested track changes as an important feature for a reviewing tool.

Furthermore, we were interested to know how often do the interviewees required converting a file from one type to another type during authoring a scientific writing. Thus we asked the following question:

- *Have you ever needed to convert files from one authoring tool's standard format to another format (e.g. from *.docx to *.odt)?*

Figure 2-11: Opinions of the interviewees about the most important features for a specialized reviewing software tool

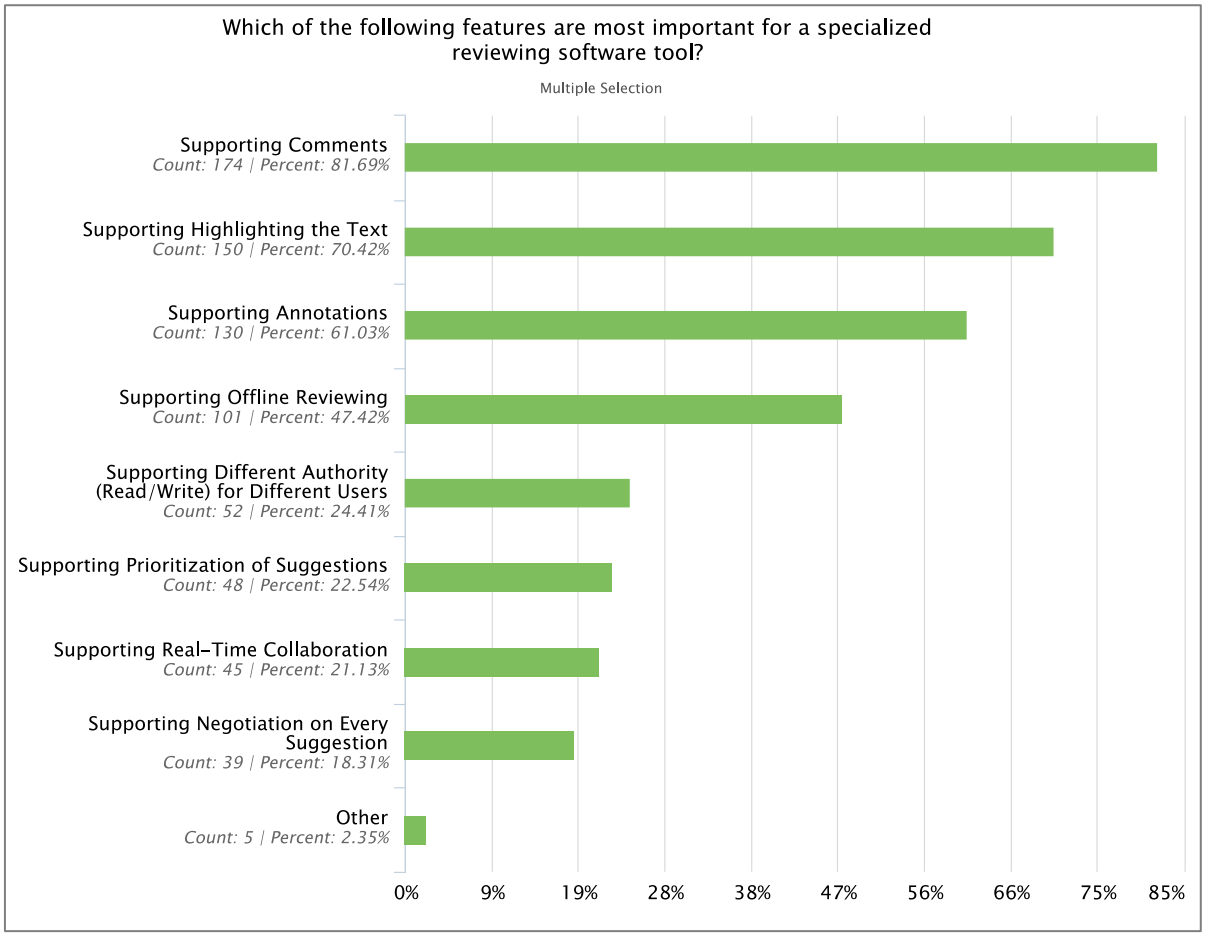


Figure 2-12 illustrates the importance of supporting other formats as importing (exporting) option for authoring tools. Definitely, it cannot be expected everybody use just the same product for editing documents. Some products are expensive and all people cannot access them, and some other people used to use some especial software and have a resistance about switching to some other products. Thus, it is definitely important for the popularity of an authoring software to support dominant file types with importing and exporting options.

Finally, we were interested to know about the problems that our interviewees may be faced when they used a converter software for converting a document source file to another type. We asked them:

- *When converting between two different file standards, what was the biggest problem that you encountered?*

We classified the answers in ten groups. The classified result has been demonstrated in Figure 2-13. It can be seen that most complaints are about missing formatting. Another problem is missing of the hierarchical text like nested lists. It is interesting that many people had a

confusing problem in working with converter tools or finding a suitable tool. Finally, missing of the graphical objects was the fourth important reported problem.

Figure 2-12: Percentage of the people who required converting a document file to other formats.

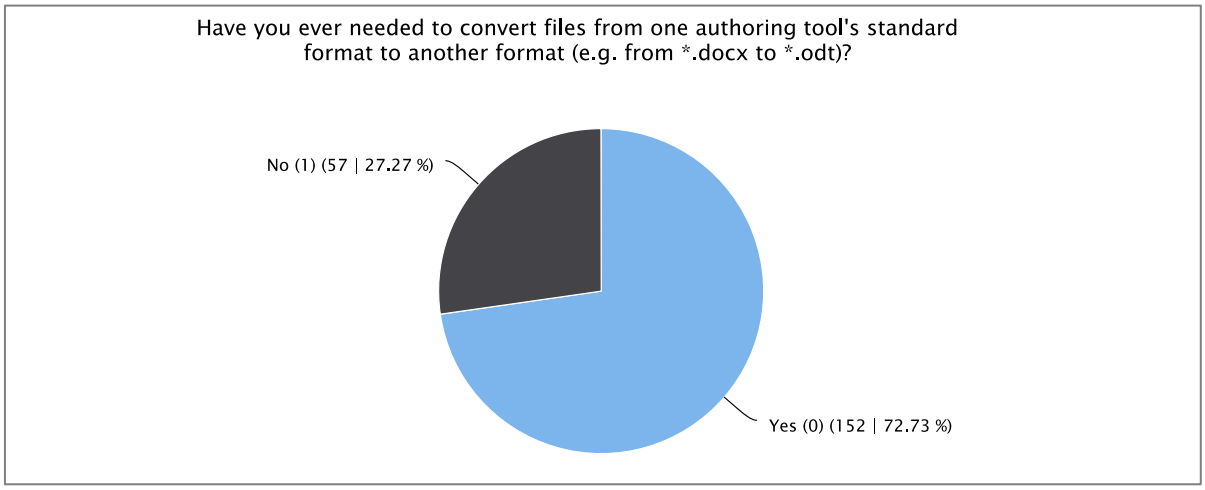
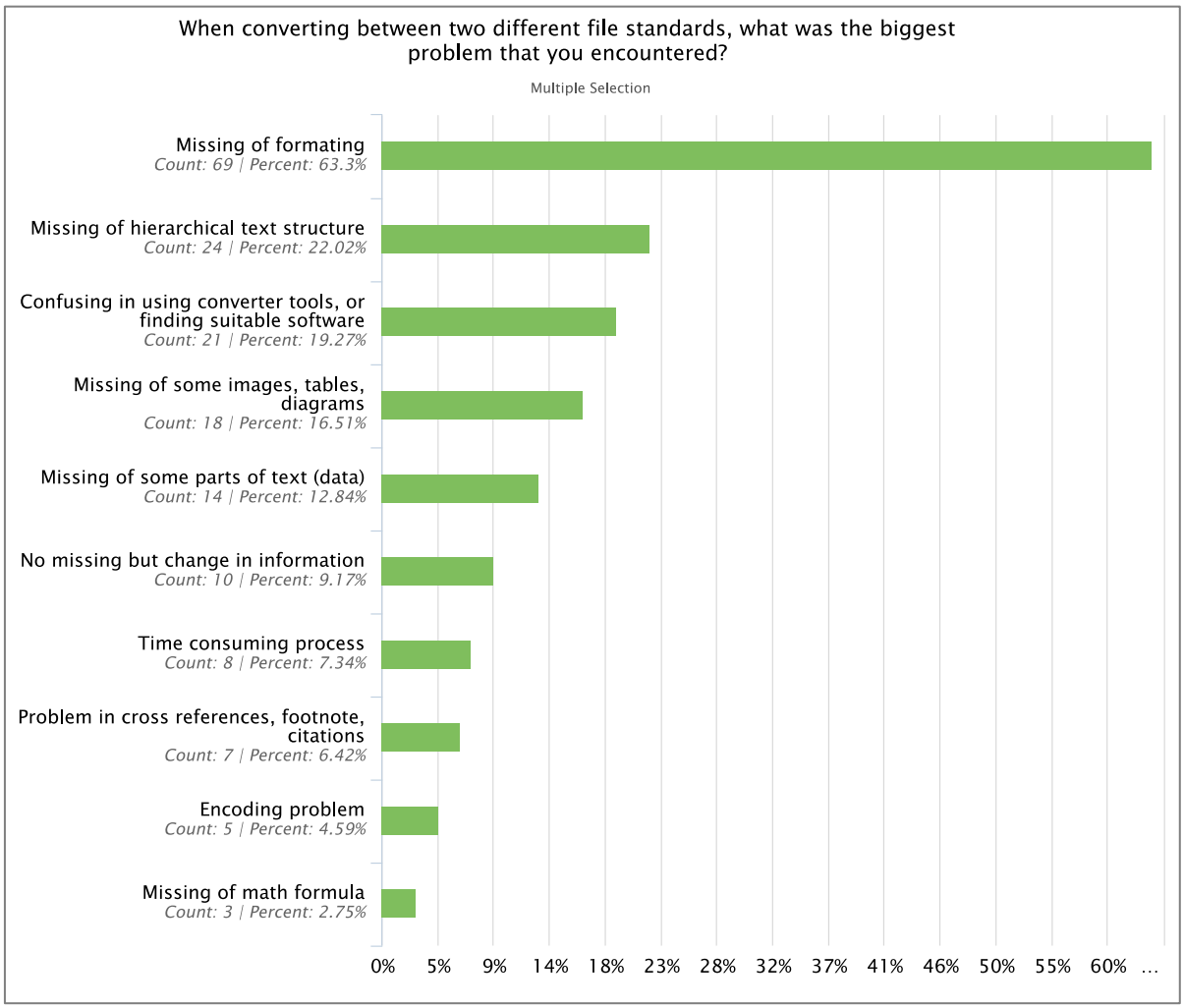


Figure 2-13: Classified result of the experiences of converting document source files



Chapter 3 - State of the Art

Popular features in scientific document files

If an authoring tool aims to be used for scientific purposes, requires supporting particular facilities and features. Thus, we can classify authoring tools to scientific usable and scientific unusable or less usable ones. For doing such kind of classification, we looked into a collection of 27 original source files of real scientific papers. Fifteen of those papers had been submitted to Elpub 2016[8], and can be accessed from OSCOSS[9] project's repository[10]. The remaining files were provided by ten individual researchers and we were not given the permission for publishing their source files beside of this thesis. As a significant number of the scientific journals accept the DOCX format as their default format, as well, based on the result of our online survey, the most popular authoring software between scientists is MS Word. Therefore, it can be concluded that the DOCX format is the most popular format file for publishing scientific papers. Therefore, we preferred to investigate the DOCX format files rather than other formats.

In our investigation on that collection of DOCX files, we counted different features that had appeared in those documents. We used (DOCX STANDARD SCIENTIFIC STYLE, 2016) resource to list standard features of scientific papers and then looked for those features in those 27 DOCX files.

Table 3-1 illustrates the total number of the usages and the total number of the correct usage of these features. The important point related to the considered features in a document is that, if they have been created in a correct way, they can be recognized correctly by converters and may be reflected correctly in the converted formats. For example, if the title of a document just has been created by setting the font size at 20 points, it can be recognized by a human that which part of the text is its title, it is not recognizable by a machine easily. Moreover, when a file is converted to another format, especially to the PDF format, usually converters can transfer

---

[8] More information about Elpub conference can be found here: http://meetings.copernicus.org/elpub2016/

[9] OSCOSS is a project that aimed to design a web-based authoring tool that authors more focused on content in a collaborative environment and style applied to the paper based on the standard style of a conference or journal. More information about the OSCOSS project can be found here: http://eis.iai.uni-bonn.de/Projects/OSCOSS.html

[10] The source files of the DOCX files which were used during our survey on the real scientific papers can be found freely in the following repository: https://github.com/OSCOSS/testdata_mda

these metadata to the destination format. These metadata can be helpful for search engines when they want to index a paper.

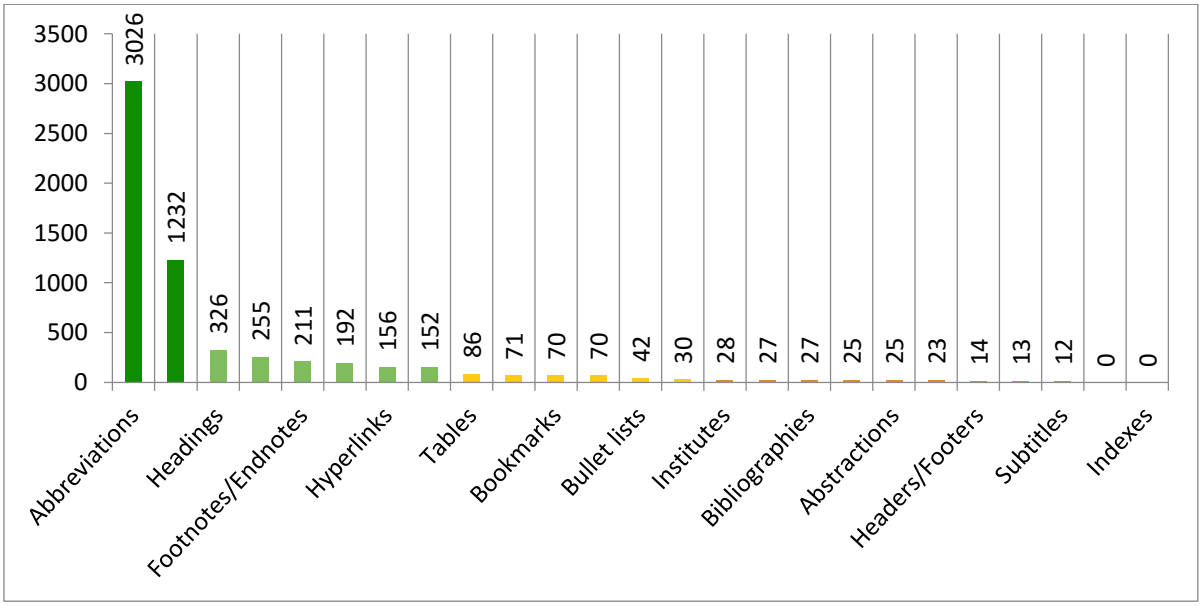
Table 3-1: Comparing used features in 27 scientific papers

Row#	Feature name	Usage	Correct Usage	Percentage of correctness
1	Title	27	8	29.63%
2	Subtitle	12	1	8.33%
3	Authors	25	5	20.00%
4	Institution	28	5	17.86%
5	Abstraction	25	5	20.00%
6	Keywords	13	1	7.69%
7	Date	0	0	0.00%
8	Headers/Footers	14	14	100.00%
9	Footnotes/Endnotes	211	208	98.58%
10	Bookmarks	70	70	100.00%
11	Cross-References	30	30	100.00%
12	Page number	23	22	95.65%
13	Hyperlinks	156	78	50.00%
14	Tables	86	83	96.51%
15	Figures	70	62	88.57%
16	Equations	71	42	59.15%
17	Captions	192	44	22.92%
18	Citations	1232	226	18.34%
19	Bibliographies	27	10	37.04%
20	Stylings	152	135	88.82%
21	Indexes	0	0	0.00%
22	Bullet lists	42	42	100.00%
23	Numbered lists	255	168	65.88%
24	Headings	326	152	46.63%
25	Abbreviations	3026	3026	100.00%

Furthermore, it can be interesting to know how this data was collected and which parameters were considered during the investigation. For example, for items that usually appear on the first page of the document (i.e. the cover page), same as title, subtitle, author(s), institution, abstract, keywords and date, if the item was created in a correct way (i.e. author used *INSERT>Explore Quick Parts>Field…* or he or she assigned suitable Style to the item) then that item counted as the *"Correct Usage"* as well. Further information related to the used methods for collecting data of the other features and more detailed information related to each document can be found in Appendix B.

Figure 3-1 illustrates which features are more used in the scientific papers. It can be seen that abbreviations and the citations are the two most dominant features that respectively appears 3026 and 1232 times in considered DOCX files. It means each document rarely had 112 and 45 abbreviations and citations respectively. The next category includes headings, numbered lists, footnotes, captions, hyperlinks and styles that appeared between 6 to 12 times in each document. The third group consists of tables, equations, bookmarks, figures, bullet lists, and cross-references which appeared almost 3 times in each document. The next group includes institutes, titles, bibliographies, authors, abstract, page numbers, headers/footers, keywords and subtitles which appeared almost in all or half of the 27 documents at least one time. The last group is the remaining features (i.e. dates and indexes) that were not seen at all.

Figure 3-1: Most used features in 27 scientific papers



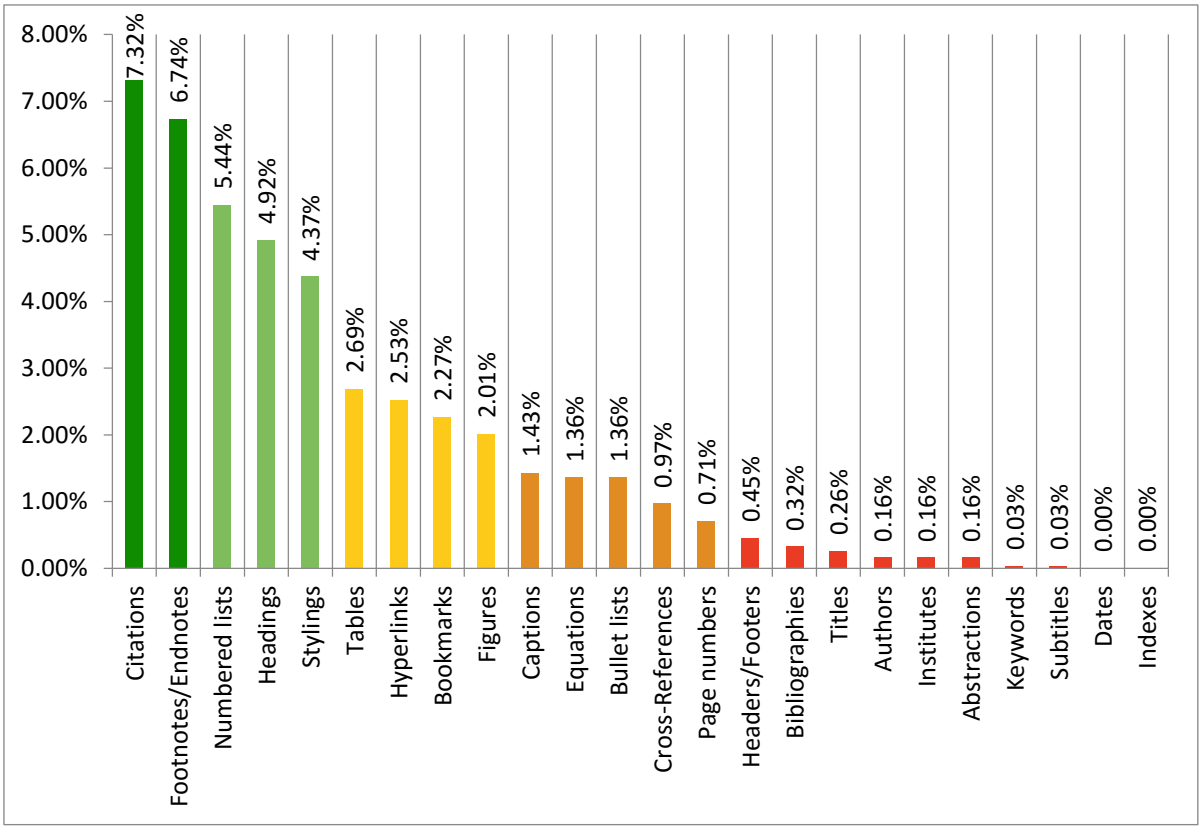
It is important to know the ratio that scientists know to create each of these features in a correct way? For example, if they entered a citation, to what extent do they use the '*Insert Citation'* facility of the authoring tools? We cannot use percentages inside of the last column of directly. The reason is that if some of the authors used one specific feature 100% correctly while that feature used in very rare cases, it cannot be compared with one other feature which was used 50 times more, however, with 90% correctness. Thus, we require calculating a weighted percentage based on the number of usages. It has been considered in Equation 3-1.

Equation 3-1: Calculation of the weighted percentages of the DOCX files' features

$$Weighted\ Percentage = \frac{Number\ of\ the\ Correct\ Usages}{Number\ of the\ Usages\ of\ Feature} \times 100 \times \frac{Number\ of\ the\ Usages\ of\ Feature}{Total\ Number\ of\ the\ Usages\ of\ All\ Feature}$$

Figure 3-2 illustrates the calculated numbers based on Equation 3-1. One important fact related to DOCX format is that there is no specific facility to tag a word as an abbreviation while it is possible in LaTeX, thus, we considered all the abbreviations as a correctly used feature, and we overlooked that feature in Figure 3-2.

Figure 3-2: Percentage of the correct usage of the well-known features of the DOCX format file.

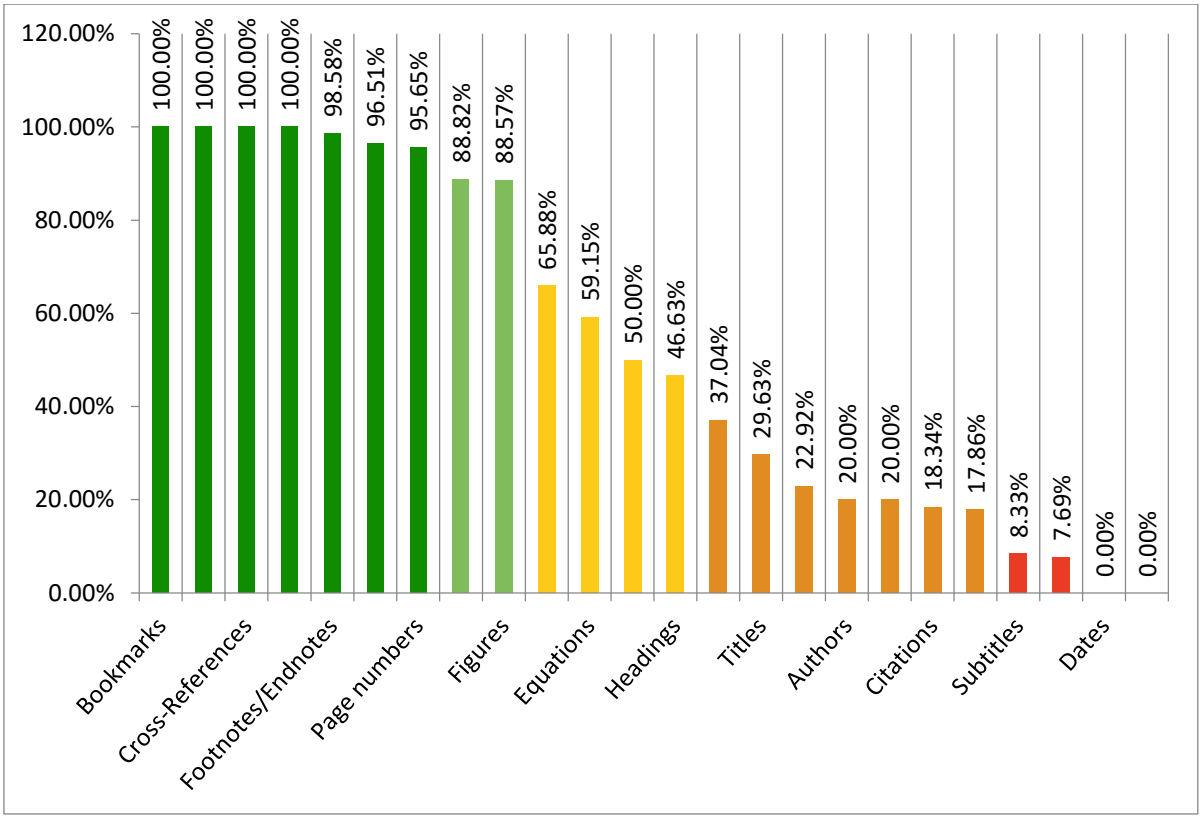


Five different categories of features can be recognized in Figure 3-3. For example, citations and footnotes/endnotes are not only used more than other features, but also scientists have better knowledge of how to insert a correct citation in comparison to a feature like the captions. However, we have to consider each paper can have just one title and not more, while normally a scientific paper is full of citations. Thus, we have to take a side look at the pure percentages of the correct usages of the features as well.

Now we can judge in a better way by consideration of both recent charts (i.e. Figure 3-2 and Figure 3-3). For example, it can be concluded that in the most cases the authors don't know how to insert the title of their paper correctly in a DOCX file because just 30% of them did it

correctly[11]. Furthermore, it can be seen that not only very rare cases added keywords to their papers but also even that small group of authors didn't know how to do that correctly. While in the most cases, that group of authors knew how to insert Footnotes. There is a clear reason for this difference. In fact, it returns to the way MS Word's menu has been designed and it is more a software engineering reason.

Figure 3-3: Pure percentages of the correct usages of the DOCX files' features in the investigated real scientific papers



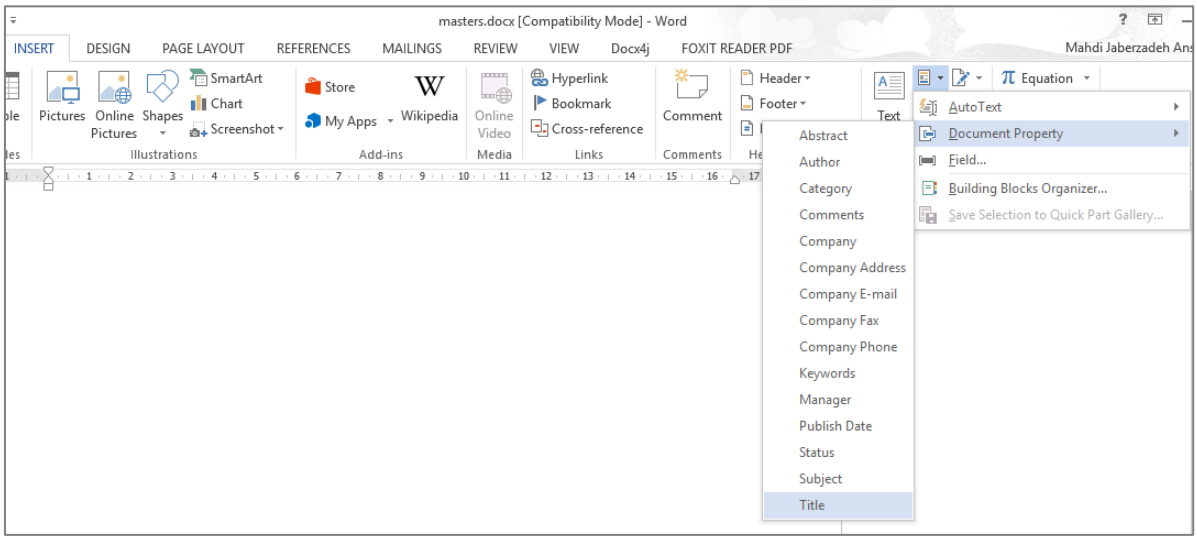
It can be seen the features that have an icon on the ribbon and a short text near to that icon, have a bigger correct usage numbers rather than those features that are hidden inside of some other buttons on the ribbon. For example, as it has been illustrated in Figure 3-4 for inserting the title of the document in a correct way someone has to do these steps in MS Word 2013:

- Go to the '*INSERT*' ribbon
- Press the '*Explore Quick Parts*' button
- Go to the '*Document Property*' submenu
- Press the '*Title*' menu

[11] Even those 30% of the authors just assigned a suitable style and they had not used the '*Insert Title'* menu.

Thus, these long list of actions can clearly explain why in the most cases, people don't know about the '*Insert Title*' facility of MS Word. While most of the users know how to insert '*Page Number*', because easily they can find its menu on the '*INSERT*' ribbon. Furthermore, as much as a menu placed near the latest ribbons then the chance for knowing that facility decreases.

Figure 3-4: The correct way of inserting the title of a document in MS Word



Authoring and Reviewing Software Tools

Our third survey was on a subset of authoring and reviewing software tools. We were interested to know how much each of these authoring and reviewing tools supports features that can be detected in a scientific writing. We considered a combination of the desktop-based and web-based authoring and reviewing tools. We asked a group of ten masters and Ph.D. students to work with these tools and assign a score between zeros to one for each feature of each tool. Then we calculated the average and rounded up the scores for each feature. Score 1.0 demonstrates they were satisfied 100%. The aggregated result is illustrated in Table 3-2. As the first result, it is helpful to know the total score of each authoring software, thus, it can be compared with the result of our online survey. The total score for each tool demonstrates in Figure 3-6.

It is notable that this group of students who asked them to participate in our investigation about authoring software tools didn't participate in our online survey and didn't know anything about our online survey or its result. In fact, we wanted to have this chance to be able to compare the idea of two different group of statistical society in our investigations.

Table 3-2: Supporting of scientific documents' features in a subset of authoring tools

Supporting Feature → / Name of the software ↓	Goggle Docs	MS Word 2010	MS Word Online	LibreOffice	OpenOffice	Overleaf	TeX/LaTeX	ShareLaTeX	Mathematica	Mathematica Online	Authorea	Lyx	EtherPad	WPS Office Writer	Zoho Doc
Title	0.5	1	0.5	1	1	1	1	1	1	1	1	1	0	1	0.5
Subtitle	0.5	1	0.5	1	1	1	1	1	1	1	0	1	0	1	0.5
Authors	0	1	0.5	1	1	1	1	1	0	0	1	1	0	1	0
Institute	0	1	0.5	1	1	1	1	1	0	0	0	1	0	1	0
Abstraction	0	1	0.5	0	0	1	1	1	0	0	0	1	0	1	0
Keyword	0	1	0.5	1	1	1	1	1	0	0	0	1	0	1	0
Date	0	1	0	1	1	1	1	1	1	1	0	1	0	1	0
Header, Footer	1	1	1	1	1	1	1	1	1	0	0	1	0	1	1
Footnote, Endnotes	1	1	1	1	1	1	1	1	0	0	0	1	0	1	1
Bookmark	1	1	0	1	1	1	1	1	1	0	0	1	0	1	1
Cross Reference	1	1	0	1	1	1	1	1	1	0	0	1	0	1	1
Pagination	0.5	1	1	1	1	0	0	0	0	0	0	0	1	1	0.5
Hyperlink	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
Abbreviation	0	0	0	0	0	1	1	1	0	0	0	1	0	0	0
Table	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
Figure, image	1	1	1	1	1	1	1	1	1	0.5	1	1	0	1	1
Image editing tools	1	1	0.3	1	1	0	0	0	1	0	0	0	0	1	0.4
Drawing	1	1	0	1	1	1	1	1	1	0	1	1	0	1	1
Equation	1	1	0	1	1	1	1	1	1	1	0	1	0	1	1
Special characters	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1
Caption	0.5	1	0	1	1	1	1	1	1	1	1	1	0	1	0
Quotation	0.5	1	1	1	1	1	1	1	0	0	1	1	0	1	1
Citation	0.5	1	0	1	1	1	1	1	1	0	1	1	0	0	0
Bibliography	0	1	0	1	1	1	1	1	1	0	1	1	0	0	0
Styling	0.8	1	0.2	1	1	1	1	1	0.5	0.5	1	1	0.2	1	1
Table of Content	1	1	0	1	1	1	1	1	0.2	0	0	1	0	1	1
Bullet list	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Numbered list	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
Heading	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Importing files	1	1	0.5	1	1	0	0	0	0	0	1	0	0.2	1	1
Exporting files	0.8	0.8	0.6	0.9	0.9	1	1	1	1	0.1	1	1	1	1	0.8

Table 3-2: Supporting of scientific documents' features in a subset of authoring tools

Supporting Feature → / Name of the software →	Goggle Doc	MS Word 2010	MS Word Online	LibreOffice	OpenOffice	Overleaf	TeX/LaTeX	ShareLaTeX	Mathematica	Mathematica Online	Authorea	Lyx	EtherPad	WPS Office Writer	Zoho Doc
DOCX export (less problems)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
DOCX import (less problems)	0.8	1	0.9	0	0	0	0	0	0	0	1	0	0	0	1
Real-time collaboration	1	0	1	0	0	0.8	0	1	0	1	1	0	1	0	1
Offline collaboration	0	0.5	0	1	1	1	1	1	0	0	1	1	0	0.5	0
Versioning	1	1	0.5	1	1	0.5	0	1	1	1	1	1	1	1	1
Comments	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
Negotiation on comments	1	1	1	1	1	1	0	0	0	0	1	0	1	0	1
Suggestions (Add, Delete, Replace)	1	1	0	1	1	0	0	0	0	0	0	0	1	1	1
Accept/Reject Suggestions	1	1	0	1	1	0	0	0	0	0	0	0	1	1	1
Highlighting	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1
Annotation	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Prioritizing Suggestions	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Classification of Suggestions	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Double Column	0	1	0	1	1	1	1	1	0	0	0	1	0	1	0
Redo, undo	1	1	1	1	1	1	0.8	0	1	0.5	0	0.8	1	1	1
Spell Checker	1	1	1	1	1	1	1	1	0	0	0	1	0	1	1
Grammar Checker	0	0.3	1	1	1	0	0	0.5	0	0	0	0	0	0.3	0
Bold, Italic, Underline	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Support Right to left	1	1	1	1	1	0	0	0	1	1	0	0	1	1	1
WYSIWYG	1	1	1	1	1	0.1	0	0	0.5	0.5	1	1	1	1	1
Total Score	33.6	44.6	27.1	42.9	42.9	36.4	33.8	35.5	26.2	16.1	24	35.8	15.4	39.8	32.7

Figure 3-5: Percentage of the supporting features by the authoring tools

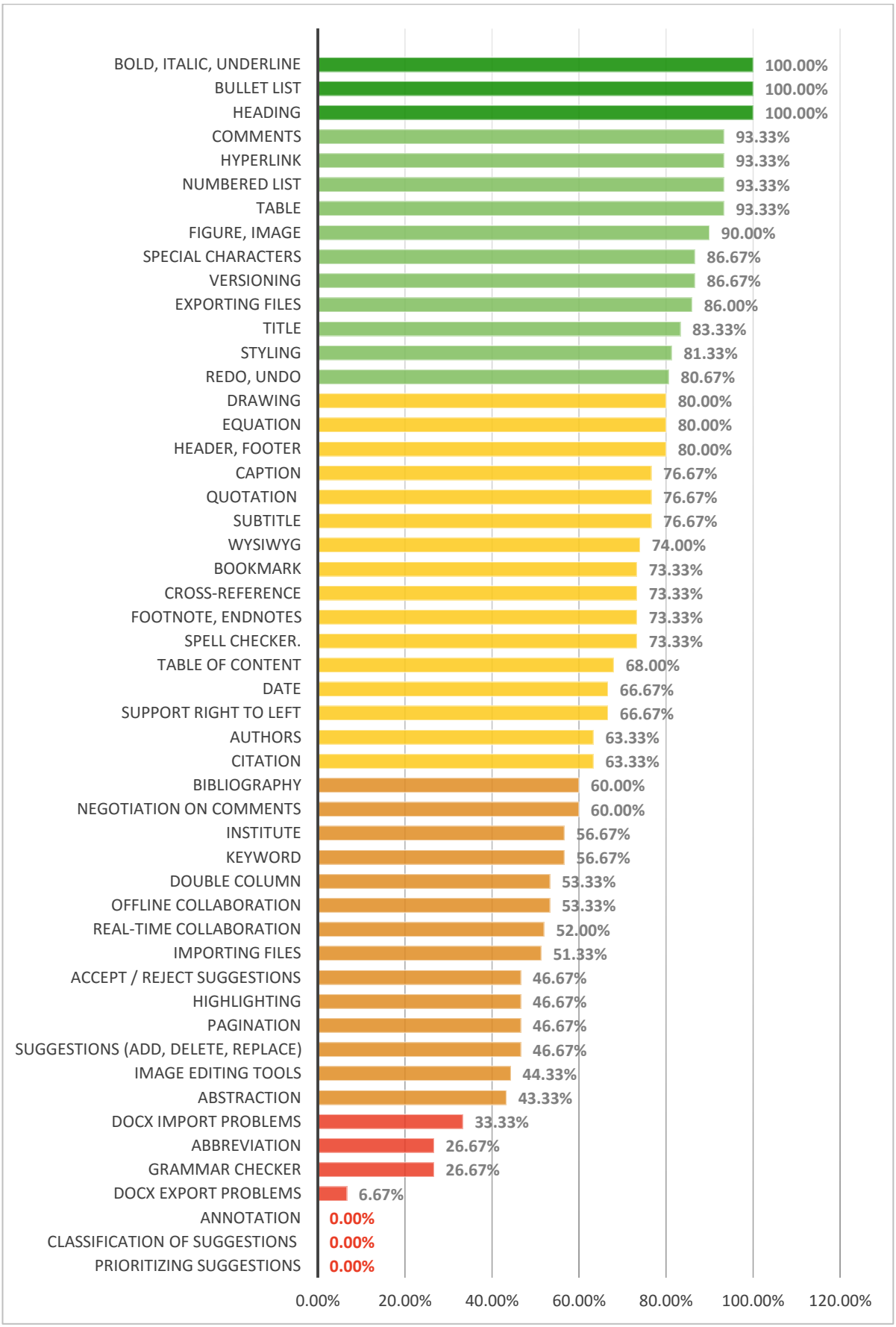
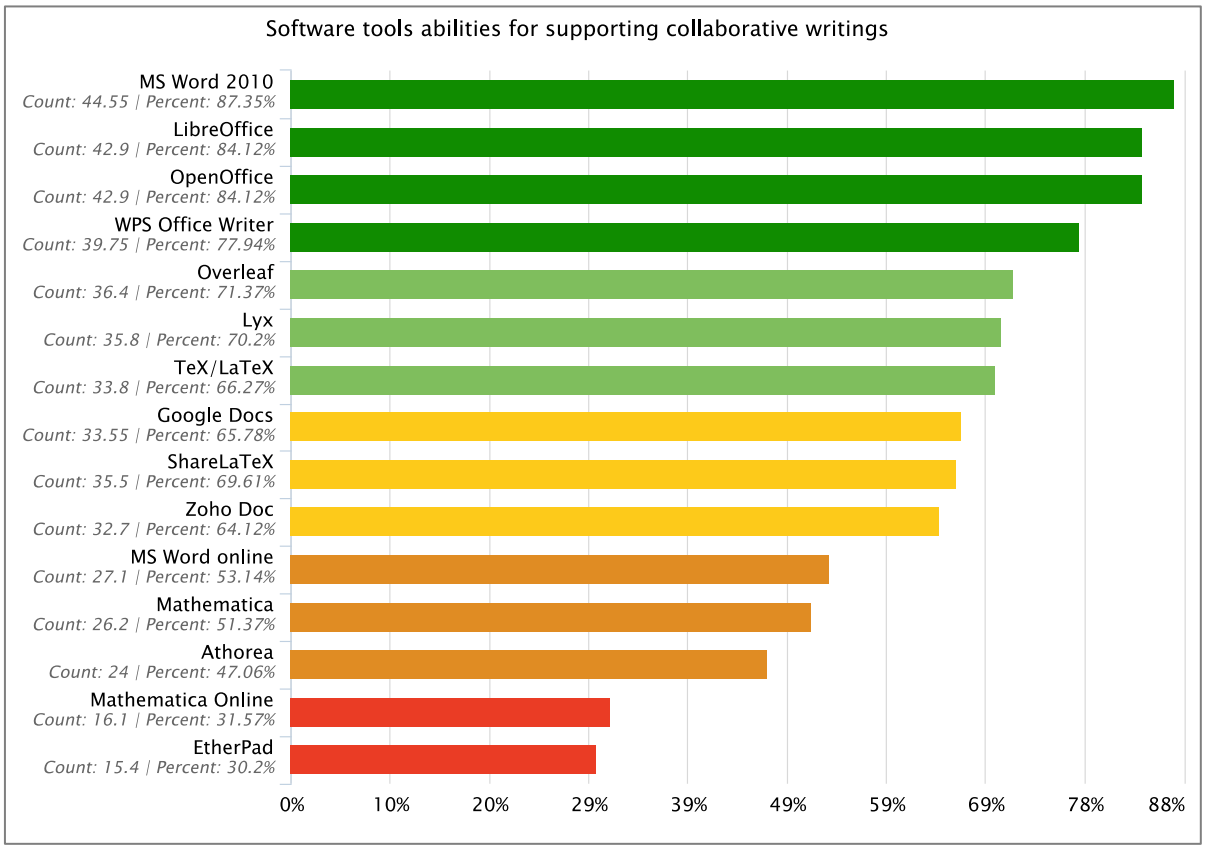


Figure 3-6: Software tools' scores for supporting academic writings



Another interesting fact is how much each of the features, is supported in these authoring tools. We were looking forward to knowing which features are more supported in authoring tools and which features require spending much more time for supporting them by the developing companies. The result is illustrated in Figure 3-5.

It can be seen the features for making the bold or italic text, underlining, heading and bullet list are exist in all authoring tools. An important point is that almost all authoring tools except Mathematica Online, support commenting. It means almost all of the major authoring tools have a reviewing strategy as well. EtherPad is the most unpopular authoring tools for scientific writings, however, despite the limitations of features in EtherPad, it has some properties that make it especial:

- First of all, it was acquired by Google and its code was published as open source in December 2009 (Pant, 2015). The source code of EtherPad can be a desirable starting point for developing new collaborative authoring and reviewing tools.
- Second, it has been written in JavaScript entirely and its server is based on Node.js. As Node.js supports stream connections between client and server, it seems the most

popular technology for providing the real-time connection is Node.js. Google Docs uses Node.js to provide real-time collaboration as well.

- It designed to support not only collaborative editing but also collaborative reviewing. EtherPad uses JSON as an internal model for the document. However, not the simple JSON. It stores JSON objects inside of a table of the SQLite database. Using a database for storing a document brings the ability for supporting versioning and tracking the changes.

It can be seen 86% of these tools support exporting to other formats, however, 51.33% have the ability for importing files from other formats. The result for supporting the importing and exporting of the DOCX format does not follow this pattern. It means most of the authoring tools can import the DOCX format in a better way in comparison to the time that they want to export it. It is interesting that even Microsoft Word Online does not support all features of the DOCX format when it wants to create a DOCX file.

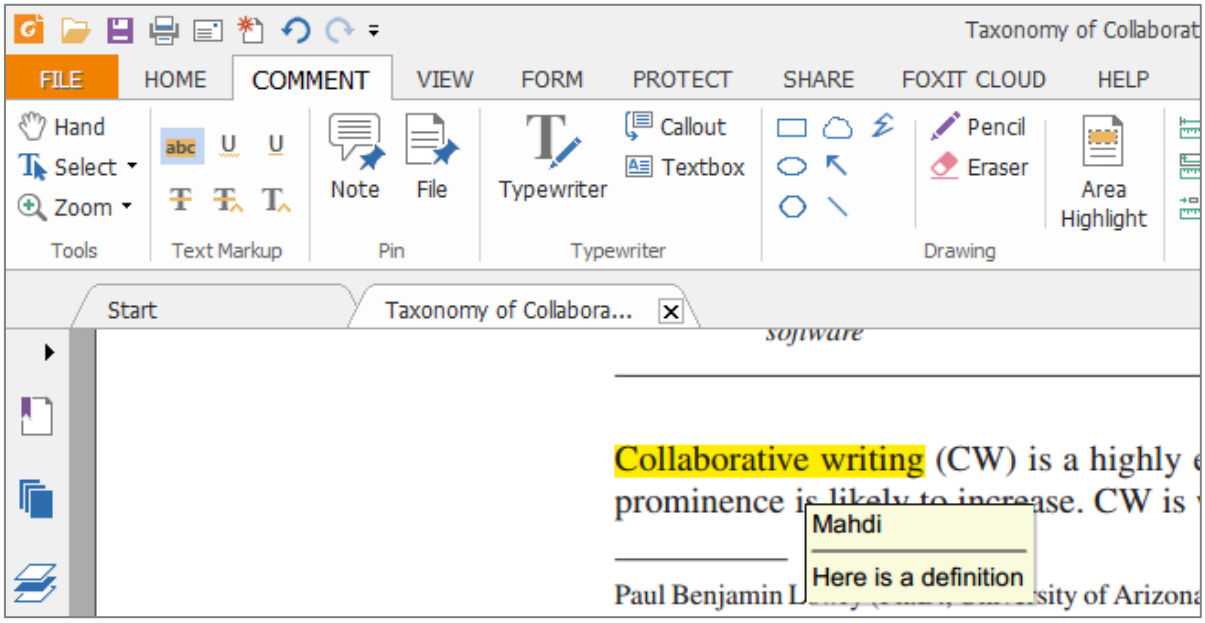
Clearly, the frailty of all of these authoring and reviewing tools is related to their support for reviewing of documents. It can be seen ShareLaTeX has the best grammar checking tool, while MS Word in the desktop versions and WPS Office Writer are the only two other authoring software which has a grammar checker, however, rarely weaker than ShareLaTeX as it uses a third-party API for providing this feature.

None of these software applications support annotation like what the PDF readers provide. Figure 3-7 illustrates a sample of annotation in PDF files. Annotations can be covered by comments, however, if a reviewing tool supports annotation separately, it can have its own usability in some cases. For example, in a collaborative environment, annotations can be used for authors' notes and reminders while the comments can be used for reporting some higher level of mistakes, or retain the comment facility for reviewers.

Unfortunately, it is not possible to tag the comments and classify the comments in the considered subset of authoring and reviewing tools. This classification can be accomplished with a search tool that could list different type of comments and help the authors and reviewers to act in a more efficient way. For example, it may happen that someone wants to classify the comments to the grammar related, semantic related and content related. Currently, this classification is not possible in any of these major authoring and reviewing tools (except Pleasereview), while it was one of the suggestions that people mentioned more than five times in our online survey. Finally, even if the classification of comments and suggestion is implemented, it may happen that someone requires considering comments and suggestions based on some order or priority. It may happen that we detect some errors that have to be

corrected in the first version, but also it may happen that some suggestions can be postponed to later versions of a document.

Figure 3-7: A snapshot of an annotation in the Foxit Reader for a PDF file



Another tool that is helpful for reviewing a document is the suggestion tool. Suggestion tool is known with "Track Changes" menu in MS Word. Figure 3-9 illustrates the suggestion tool of MS Word. A same tool exists in the Google Docs. The key feature related to suggestion tool is that it retains the old situation of each change near its new situation simultaneously. Usually, authors can make the final decision related to each change. Authors can accept a change or reject it, or even negotiate via comments with people who made the changes. There is a specialized web-based collaborative reviewing tool for the DOCX format file which is Pleasereview. It provides a collaborative environment that all the reviewers can see each other comments or suggested changes. The reviewer cannot change the document directly and they can just suggest some changes. It is the author who decides to accept, reject or consolidate the suggestion. Finally, after making the decision about all the suggestions, it is possible to download the final document again (PleaseTechLtd, 2014). Unfortunately, when we asked the Pleasetech Company to provide a short demo access to their product, they just provided a webinar for introducing the Pleasereview tool. Thus, we couldn't evaluate their product efficiently. Here is a snapshot[12] of the Pleasereview software in Figure 3-8.

[12] The source of Figure 3-8 is (PleaseReview™ for Document Collaboration, 2016). Figure 3-8 illustrates the categorizing of the suggestions and the situation of the document before and after the changes.

Figure 3-8: A snapshot of the Pleasereview web-based software

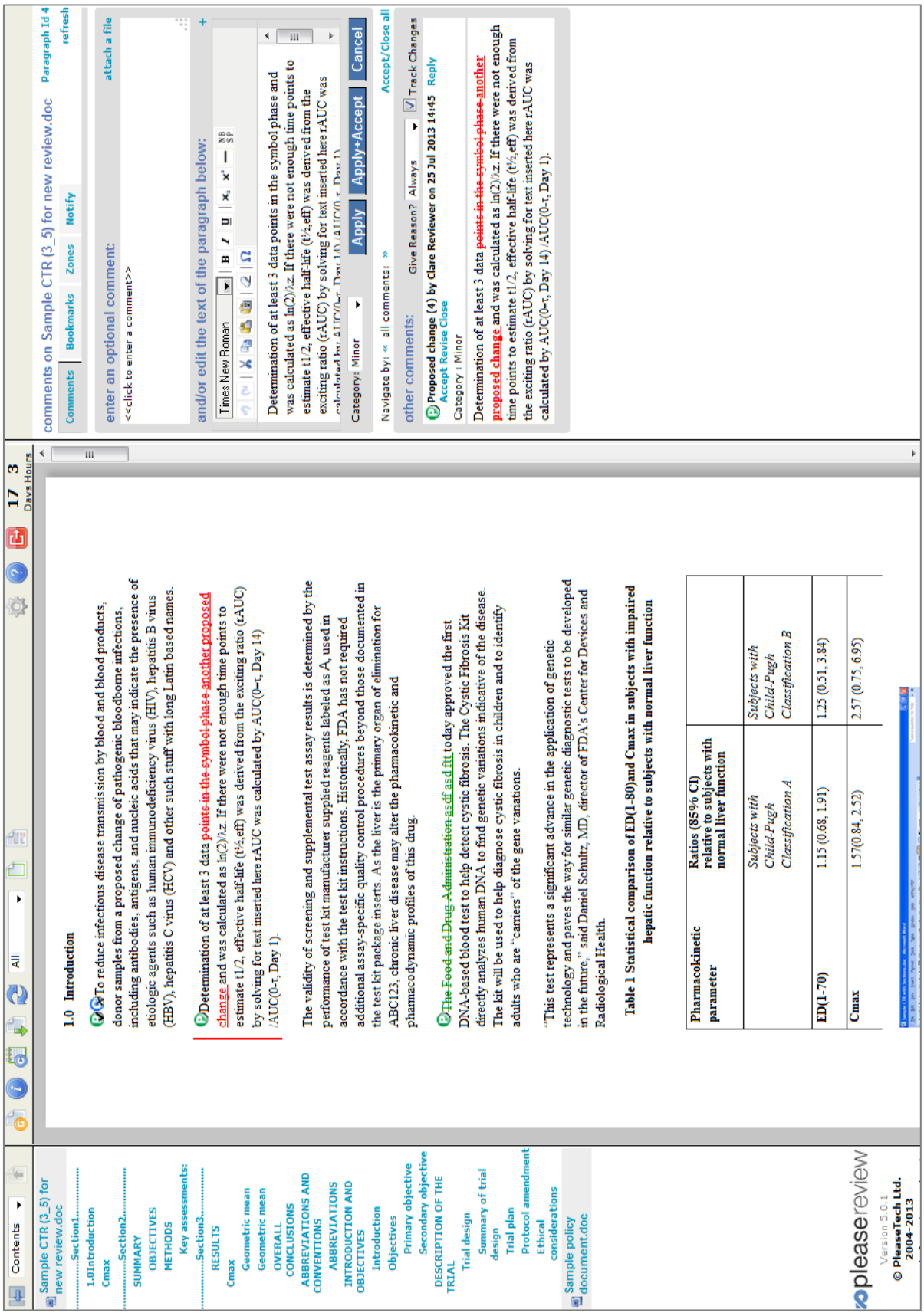
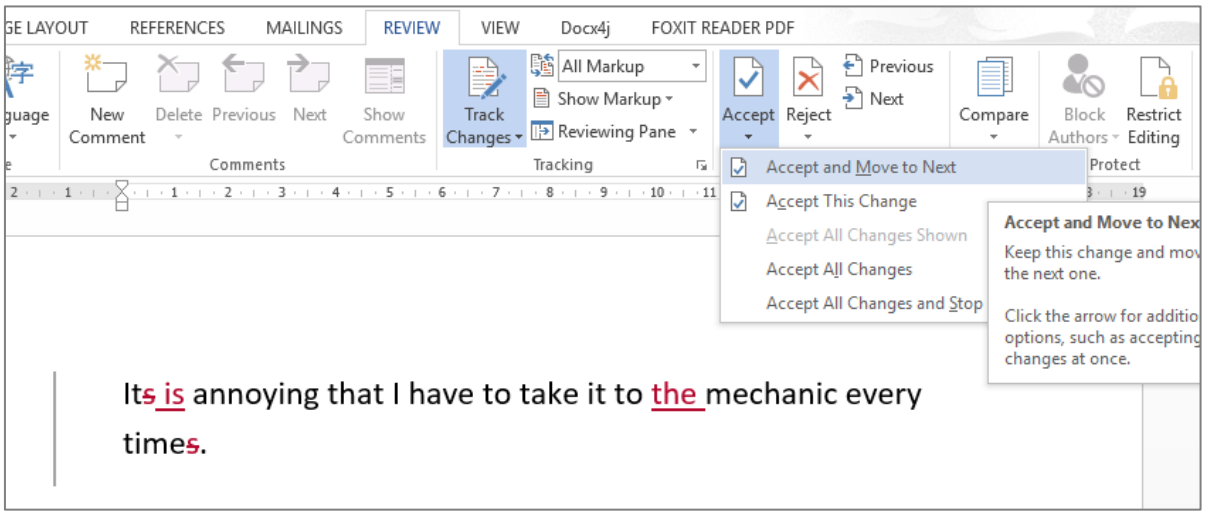
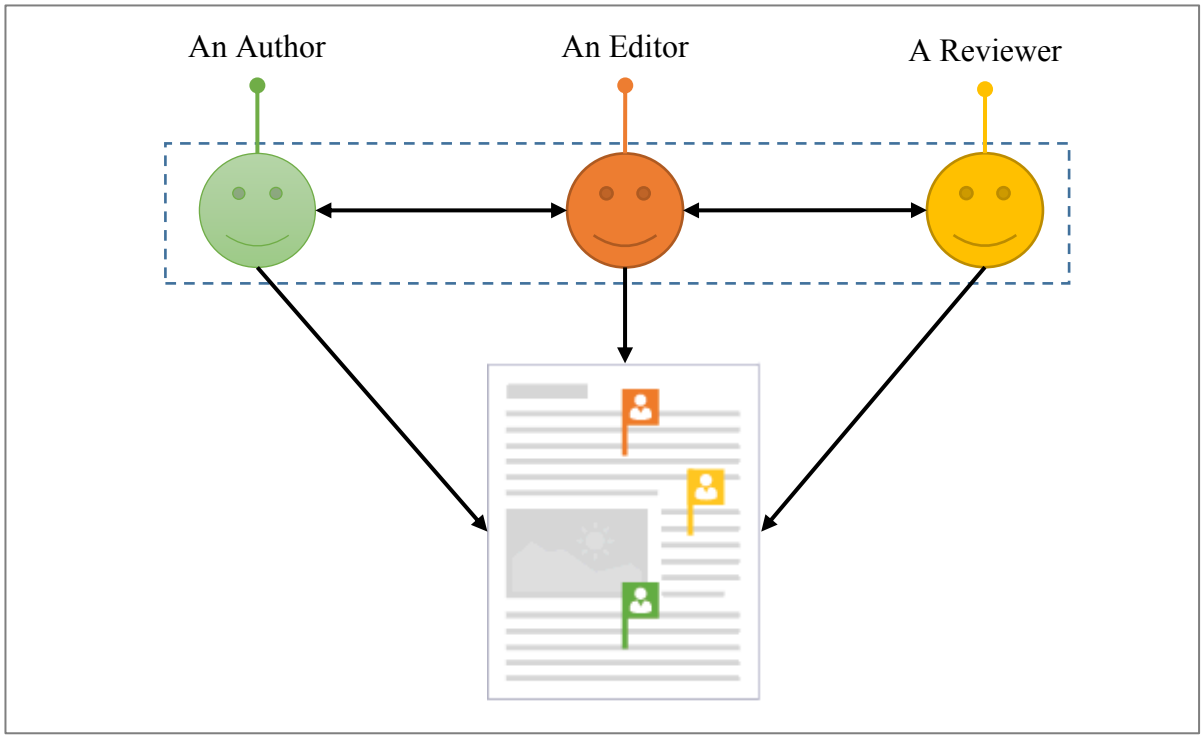


Figure 3-9: A snapshot of the "Track Changes tool" of MS Word



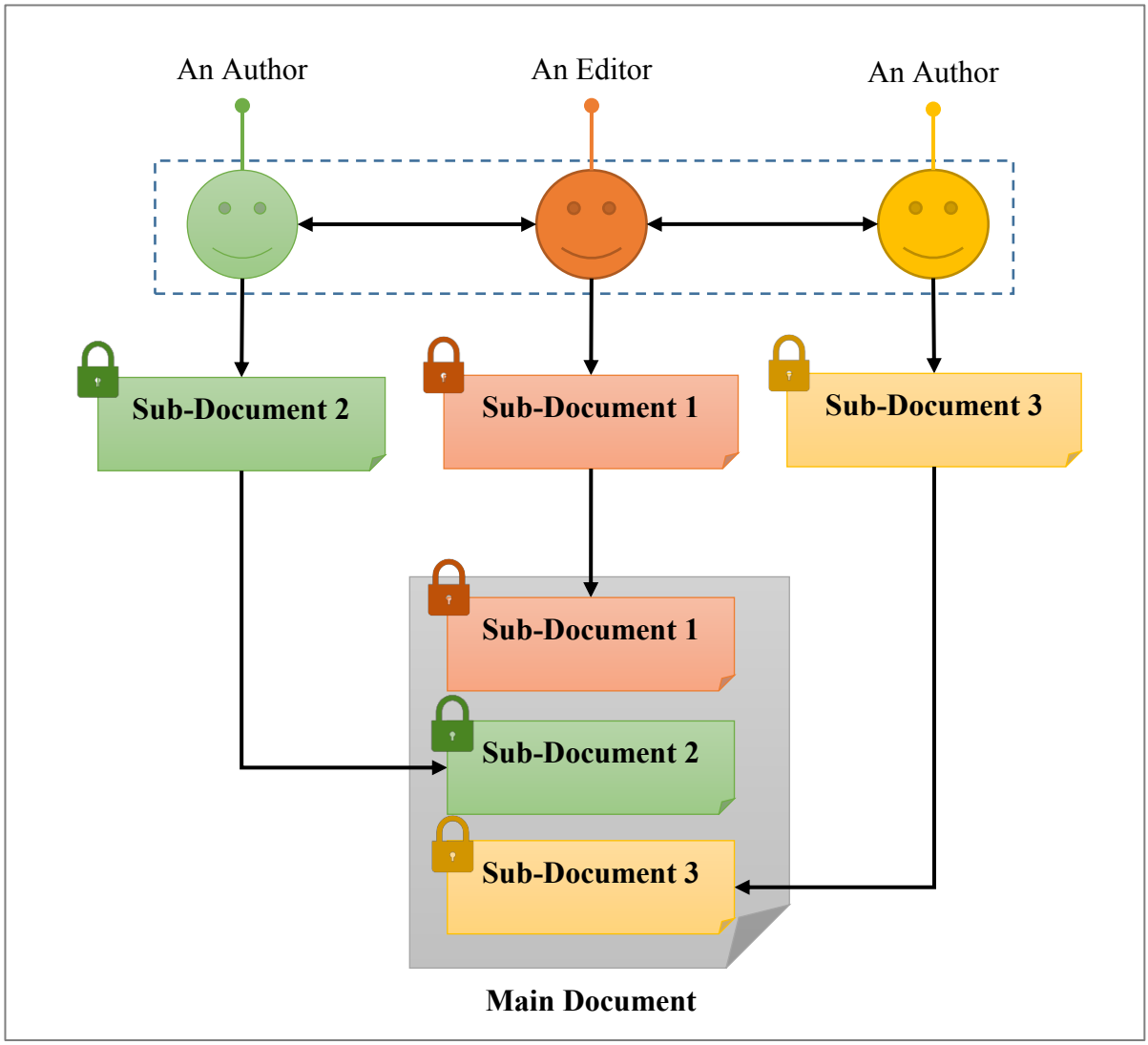
Another major issue in authoring and reviewing tools is the collaboration concept itself. There are different paradigms for collaboration on a document. The real-time collaboration can be modeled as a parallel writing. As Figure 3-10 illustrates all the authors, reviewers and editors can access the document at the same time. It doesn't matter who changes which part. It may happen that a user types something and another user delete or edit those changes. It means there is no control on the conflicts that may happen during the access by different users. In fact, it is the responsibility of the users to prevent the conflict situation or even resolve the problem that could occur during a shared access. Usually in this type of sharing the users can edit the same part simultaneously, clients send their change requests to the server, and the server changes the document based on the requests queue (fast arrive, first done). The server sends the latest changes to all clients continuously. As it is demonstrated in Figure 3-10 usually a flag indicates who is editing which part. With this method, usually users try to not work on the same paragraph simultaneously to prevent conflicts. This type of parallel writing is called stratified-division writing (Lowry, Curtis, & Lowry, 2004, pp. 78-79).

Figure 3-10: A model for the parallel writing (Stratified-Division Writing)



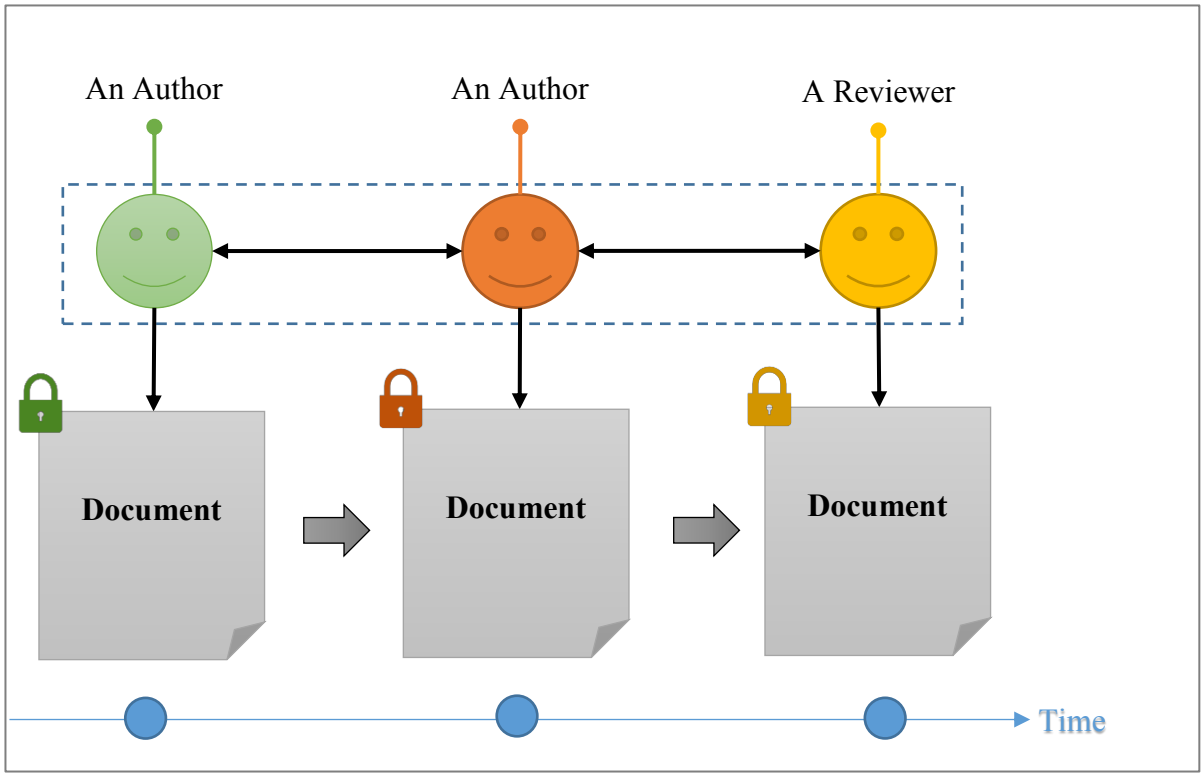
Stratified-division writing is used by most of the major web-based authoring tools like Google Docs and MS Word Online. Another type of the parallel writing is called horizontal-division writing, which has been demonstrated in Figure 3-11. The collaborative authoring and reviewing software tools that use horizontal-division writing paradigm, usually lock a part of the document when a user is editing that part. Usually, a part of a document means a paragraph of that document. The most well-known authoring and reviewing tool that uses this strategy is Authorea. Authorea supports real-time collaboration writing by locking the paragraph. In this type of authoring software, the conflict problems will not happen at all, however, some other problems like dead-connection are possible. It may happen that a user locks a paragraph, however, it loses his connection to the server (e.g. because of some technical problem) and the paragraph is left locked. Thus, other algorithms are required for checking the connections periodically and unlock the locks that exist without any connection.

Figure 3-11: Horizontal-Division Writing



Another paradigm for collaboration is sequential writing. Sometimes sequential writing is mixed up with offline writing, however, sequential writing is a sample of offline writing. In fact, in sequential writing, authors, reviewers, and editors access the document in a defined order. It can be imagined by locking the whole document. It may happen that one author starts the work and after some work, gives it to the second author. The second author can edit any part of the first author's document or add some new parts. The second author passes it to the next person after finishing his or her changes. In fact, sequential writing is common in collaborative fiction and doesn't require any especial collaboration authoring software. Sharing document in a shared file system like Dropbox can fulfill this type of collaboration.

Figure 3-12: Sequential writing

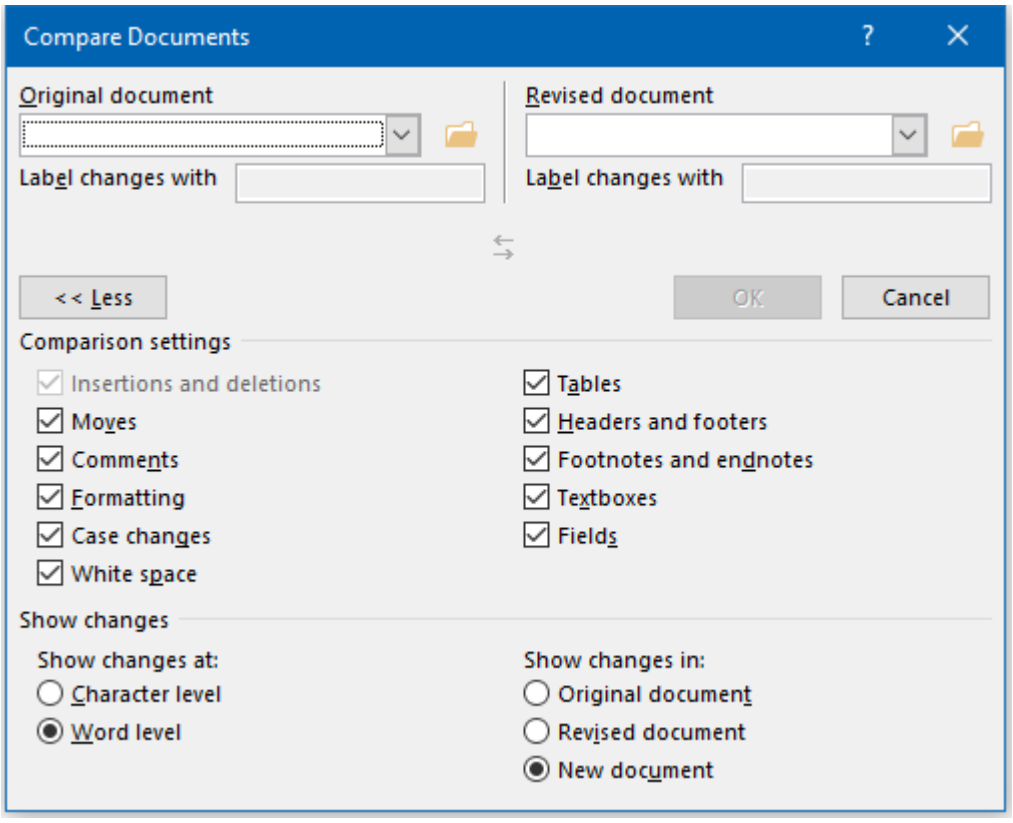


Offline collaboration is the most challenging task in collaborative authoring and reviewing software. Unfortunately, in most of the major authoring and reviewing tools, this type of the collaboration has been overlooked because of its difficulties. As we mentioned in Chapter 1 Git is a successful sample of versioning software tools that provide collaborative ability between programmers. The type of the collaboration that Git provides is offline collaboration. In fact, offline collaboration is possible in the world of programming because of two facts:

1. The source code files are simple text files and not binary files. Thus, Git algorithm requires comparing two simple text files and reports their conflicts.
2. The people who participate in a collaborative programming (i.e. programmers) know the used markup language in the project. Thus, if the Git reports some conflicts they can make a correct decision about choosing which part and how to resolve the conflict problem.

This situation can be compared with a situation that we want to use Git for a document like DOCX files. First of all, a DOCX file is a compressed file of some other XML and non-XML files. So the algorithm for comparing two versions of a DOCX file will be complicated as it requires to compare more than one file for some simple changes in WYSIWYG editor. Furthermore, if some conflicts happen, it is not possible to ask the end users to edit the XML files directly and resolve the conflicts. Therefore, the only authoring and reviewing tools that support offline collaboration, are those software tools which use LaTeX markup language like ShareLaTeX, Authorea, Lyx and Overleaf. Almost all of the web-based authoring tools that are made based on LaTeX markup language have this ability to be connected to GitHub and share the document files with other contributors. Currently, MS Word in its desktop-based versions from 2007, supports offline collaboration with its Compare menu. Figure 3-13 illustrates a snapshot of this tool in Microsoft Word 2010.

Figure 3-13: Compare tool of MS Word 2010



Chapter 4 - Implementation of the Conversion Software Tool

Document Conversion Tools

The first question that someone may ask is, why do we require document conversion tools? It has a clear answer. It is usual that someone has a particular authoring software package installed on his or her computer, however, he or she can't access the data of a document file because it is not in the required format. It is the first point that we require a conversion tool that could convert the file from one type to another.

This scenario can be more common when we want to do collaborative writing. It can happen that a group of people wants to write a paper collaboratively, however, some of them are used to using one software and some other prefer to use another software. It would be great if both of this two software support some similar format files, then the authors can make an agreement on using one of those formats. In this way, they can share the document file in a file share system like Dropbox and work on the same file collaboratively. However, what would happen if the used software tools don't support even one similar format file? Then it is the point that we require a conversion software obviously. Furthermore, it can be the case that authors use an authoring tool, however, the reviewers who are usually from outside of the authoring group use another program. Then mostly it is not possible to ask them to use the same software as authors used because mostly they have their own workflow and standards.

It can be seen most of the major authoring and reviewing software tools have some options for importing and exporting of the DOCX format file. Furthermore, there are many journals which only accept academic writing in the DOCX format. We saw in our online survey MS Word is the most popular authoring software and thus it is not unexpected that DOCX considered as the most popular format file for forming the scientific documents as well. Therefore, it seems supporting of the DOCX format file is a key feature for the popularity of an authoring tool in the realm of the scientific writing.

An advantage of the DOCX format is that its documentation is accessible freely. In fact, the DOCX format is part of the Office Open XML standard which is called OOXML. OOXML standard has been developed by Microsoft and 12 other companies and organizations for representing word-processing documents, spreadsheets, presentations and graphical charts. OOXML adopted by ECMA International as ECMA-376 in December 2006 (Standard ECMA-376, 2016). ECMA is a private international standards organization. The difference between the ISO/ICE and ECMA is that ECMA was made by the companies, while ISO/ICE was made

by the countries. ISO/ICE adopted the Open Document Format (ODF) before ECMA in 2006 to solve the document standardization crisis. This is the format that is used by LibreOffice and OpenOffice. ISO/ICE on 2nd of April 2008 approved OOXML as a new international standard for document description and processing languages (ISO, 2008). ISO/ICE 29500[13] is the identification number of this standard.

The OOXML files formats have been used as the default target files formats from Microsoft Office 2007. The biggest problem for support of OOXML by other office software suites is that there is a difference between the OOXML specification and OOXML implementation in MS Office. These differences are reflected in ISO/IEC 29500 Transitional and ISO/IEC 29500 Strict. The markup language used in ISO/IEC 29500 Strict is a subset of the markup language which is used in ISO/IEC 29500 Transitional, however, the schemes use different namespaces and are distributed separately (Preservation, 2016). Thus, it is required supporting ISO/IEC 29500 Transitional and Strict separately by software tools. For example, Microsoft Office 2010 supports reading and writing for ISO/IEC 29500 Transitional and supports reading for ISO/IEC 29500 Strict (TechNet, 2011), however, Microsoft Office 2013 supports both reading and writing of ISO/IEC 29500 Strict (TechNet, 2014).

If we overlook this problem, supporting a standard with more than 5000 pages is definitely a tough task. Thus, many libraries have been formed to support the OOXML file format. It is not so much difficult to find a library in an intended programming language that supports read and write of OOXML files. However, the negative point is that each of these libraries has its own limitations. Therefore, in cases that there is not any limitation of the programming language which the library has been developed with, it is recommended to focus on the pros and cons and abilities of the library. In fact, there are many libraries which only support almost some major parts of the OOXML standard. While most of the libraries don't speak about their limitations, it is definitely important to detect their limitations before starting the development based on those libraries.

Furthermore, it can be seen most of the major office suites support the OOXML format files. Thus, it seems supporting of the DOCX format file is a key feature for making an authoring and reviewing software tool more popular and more effective. As we discussed already, it may happen that an author requires exporting a document to the DOCX format file, makes some changes or commenting by someone else and imports it back again to the source

[13] The documentation for ISO/ICE 29500 standard is accessible from this url: http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html

format. This story even can happen in vice versa, and he or she requires converting the DOCX format file to some other format and again returns back to the DOCX format file. In such cases there are some major considerations for the end users:

1. A converter must keep the formatting of the file in both conversions. It means when the file passes this round trip it must have its formatting as much as possible.
2. A converter must convert all parts of the file correctly. It can be seen that many converters do not support some features of the document files. For example, if the document has a title object, it must be converted to the title object in the destination format and not to the simple text.
3. A converter must prevent data missing or at least make it as minimum as possible. For example, many converters do not notice the metadata. The metadata must transfer to the destination file format.

Therefore, one of the goals of this thesis is to provide a comparison between different methods that can be used for working with the DOCX format file. For this purpose, we decided to develop a converter between the DOCX format file and the FIDUS format file. This process could help us to make a better understanding of the challenges of supporting DOCX format file in an authoring software tool. The FIDUS format file is a compressed file of some JSON files and is used by Fidus Writer. Fidus Writer[14] is a new generation of web-based authoring and reviewing tools that are made for supporting collaborative writing. The idea behind Fidus Writer is that the editor focuses on the content rather than the layout, thus with the same text, it is possible to publish it in multiple ways later, as a website, as a printed book, or as an eBook (Wilm, 2016). Fidus Writer is an open source software were developed with Python and JavaScript. The reason that we selected Fidus Writer was the JSON format that is used by Fidus Writer. JSON format is the best file format for web-based authoring tools as JavaScript supports JSON. Therefore, we tried to develop a converter for converting files from FIDUS to DOCX and vice versa.

Fidus Writer is currently under developing for a bigger project which is called OSCOSS. In explanation of the goals of the OSCOSS project, it has been said that the OSCOSS team is trying to integrate existing data and publication management services into a web-based collaborative authoring environment (i.e. Fidus Writer) that publishers can set up to support all

[14] https://www.fiduswriter.org/

types of end users throughout the publication process: authors, reviewers and readers (EISGroup, 2015).

During our investigation for developing the converter, at first, we tried to create a general solution for working with DOCX files. Thus, as a start point we tried to use the Xtext and Xtend[15] to create a compiler that receives the DOCX file as a source file, compile it with some grammar rules that describes OOXML markup language, and finally, it produces the FIDUS file. However, very soon we understood that it is definitely an ambitious goal. During the development we suffered the following problems:

1. Writing a set of grammar rules for parsing the DOCX markup language with all of its exceptions was not a straight forward task. Furthermore, we wanted to create a usable converter not a tool with many limitations.
2. In the XML files, the positions of the properties of the tags are not deterministic. For example, the following headers are identical:
    - `<?xml version="1.0" encoding="UTF-8" standalone="yes"?>`
    - `<?xml encoding="UTF-8" version="1.0" standalone="yes"?>`

    This makes the grammar a little bit more complicated. Furthermore, the DOCX files created from multiple XML files. It is possible to support multiple files with the same extension in Xtext, however, it requires different grammars for each file. Different grammars for each file means we have to have some deterministic symbol at the beginning of each file to make it clear for the parser which set of grammar rules must be selected (A.H., 2013), while it can be seen the deterministic symbols in XML files are meaningless. Thus, the parser requires testing many different branches, it means parsing such set of XML files is a time-consuming task.
3. It was not only the DOCX file but also we required passing same steps for FIDUS format file as well. As it is created from a collection of JSON files. However, the negative point related to Fidus Writer is that it has been developed without any documentations or standards. That made it definitely difficult to handle this format file. Thus, we required using the trial and error method, which could make this process with many unforeseen problems and redundant works.

All of these limitations caused we conclude that not only it is not possible to create such tools in the limited time of doing a master thesis, but also if we could create a tool with the

[15] More information about the Xtext and Xtend : https://eclipse.org/Xtext/documentation/index.html

Xtext and Xtend, mostly it is not efficient enough to be used as a usable tool. Thus, we started to try other methods.

Another possibility is using the XSLT technology. The XSLT was designed to transform an XML document into another XML document, or another type of the document. While it is possible to create JSON files with the XSLT, as well, it is possible to attach linked data from other files with the XSLT, however, writing such an XSLT file that covers all or even part of the DOCX standard was a little bit messy and time-consuming work. We had to create a collection of XSLT files that reads the source files line by line and transfer them to some symbols in JSON format. In this way, any update or change in the markup language of the DOCX files could cause extra works for updating the whole XSLT files. Thus, we tried to find a solution that could have more efficiency and usability and especially its maintenance be possible. The best way was selecting a library that is the most powerful one and could be used in an open source software like Fidus Writer. Thus, we started to evaluate different libraries. In this evaluation especially we considered the abilities of the libraries in reading and writing elements of the DOCX files that are more common in the scientific papers. We selected four different programming languages and looked for some libraries that had been written in those languages. Here is the list of the selected languages and the reasons for selecting each of them:

1. **Python:** Fidus Writer has been developed by Python language. Thus, this language has the most priority for developing a backend tool to support Fidus Writer.
2. **JavaScript:** Finding a general solution with JavaScript could be used in almost all browsers and was a solution for other web-based authoring tools as well.
3. **Node.js:** As the most popular type of the server for developing a web-based tool that supports real-time connections and streams, Node.js is an interesting language.
4. **Java:** As the most popular and powerful object oriented language which is possible to run it on almost all platforms, Java was one of the best languages that are possible to use in both desktop-based and web-based tools (server side).

Table 4-1 illustrates the list of the selected libraries and their capabilities. Please note that the number of the open source libraries that could find through the internet is definitely more than this list. We just mentioned those libraries that are more brilliant and seems they used widely in different projects. Thus, we eliminated unimportant or commercial libraries from this list. Furthermore, related to the languages, maybe it is said that why we didn't consider C# or PHP and so on. In fact, we selected each of these four languages (i.e. Python, JavaScript, Node.js, and Java) as an indicator of a family of the programming languages. For

example, Node.js as the indicator of the server-side languages, or Java as a language that is used for developing desktop-based applications.

It can be seen in Table 4-1 the most powerful library for reading and writing DOCX files is the docx4j. Moreover, it is the only library that is able to manipulate the elements of the cover page like title or abstract of the document, or create and read citations and bibliography. These are not the only features that make it the best choice for working with the DOCX files, however, this library has two especial features that make it definitely especial and suitable case to be chosen:

1. The docx4j library uses JAXB to create an in-memory object representation of the DOCX file. What does it mean? JAXB stands for Java Architecture for XML Binding. JAXB can read an XML document and make an equivalent Java content tree for that XML document (unmarshalling), as well, it can create an XML document file based on the Java content trees (marshaling) (Oracle, 2016). In fact, docx4j uses JAXB to create a model of the DOCX file inside of the memory. In this way, we are able to manipulate the equivalent Java content trees directly. It means if the docx4j library has some limitations and does not support some part of the OOXML standard, we can inject the XML content directly into the JAXB wrapper. For example, we could cover the limitation of the docx4j library related to the working with math equations[16].
2. Another important and helpful feature of this library is its helper. It has a Helper add-on for MS Word. This Helper can be installed in Windows OS and would appear as an individual tab inside of MS Word's ribbon. It is possible to select a part of a document and get the help of the Helper for creating that part programmatically. For example, assume it is not possible to find a suitable example inside of the docx4j documentations for creating the hyperlinks. Just create a hyperlink inside of a DOCX document. Then select that part of the text, and press the 'Generate code' button, it will provide the code that is required for creating a sample of that element. The first method illustrates how to create the element with the help of docx4j methods, and the second method illustrates how to inject the

[16] Aspose Words is another java library for working with DOCX files that uses JAXB for parsing the DOCX files. However as it is not an open source library we didn't consider it inside of our investigation. As well, docx4j was pretty enough powerful that we didn't require using commercial libraries.

related XML code directly in JAXB wrapper (Plutext, 2016). This facility available online at the following link as well:

http://webapp.docx4java.org/OnlineDemo/PartsList.html

Figure 4-1: The docx4j Helper inside of MS Word

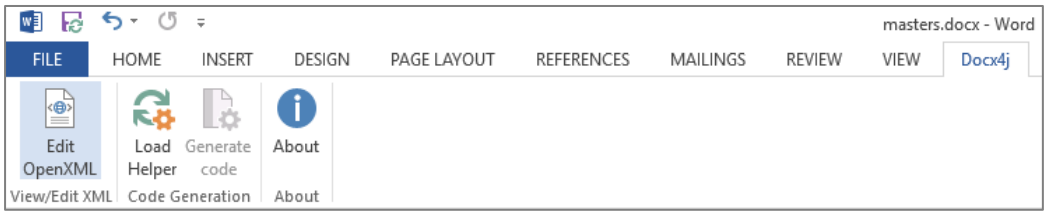
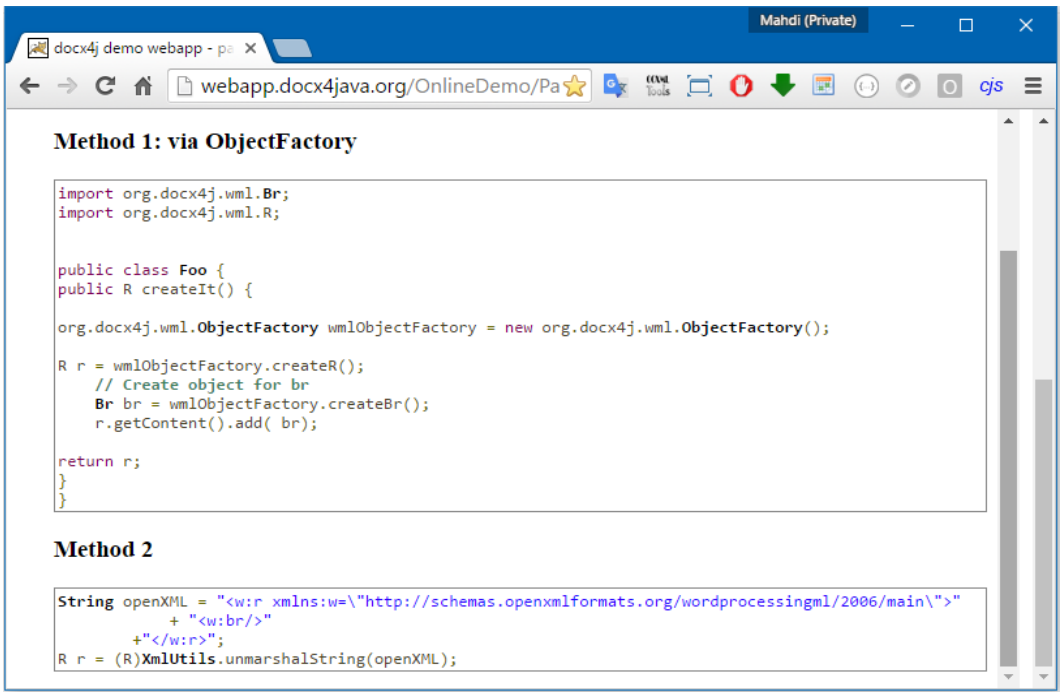


Figure 4-2: A view of the output of the docx4j Helper



These advantages were some clear reasons for selecting the docx4j library as a core for developing our converter based on. The docx4j library was created by Plutext Pty Ltd in 2008. Plutext still supports the project, however, as it is an open source project, many people contribute to developing of this project.

Table 4-1: Comparison of the libraries for working with the DOCX format file

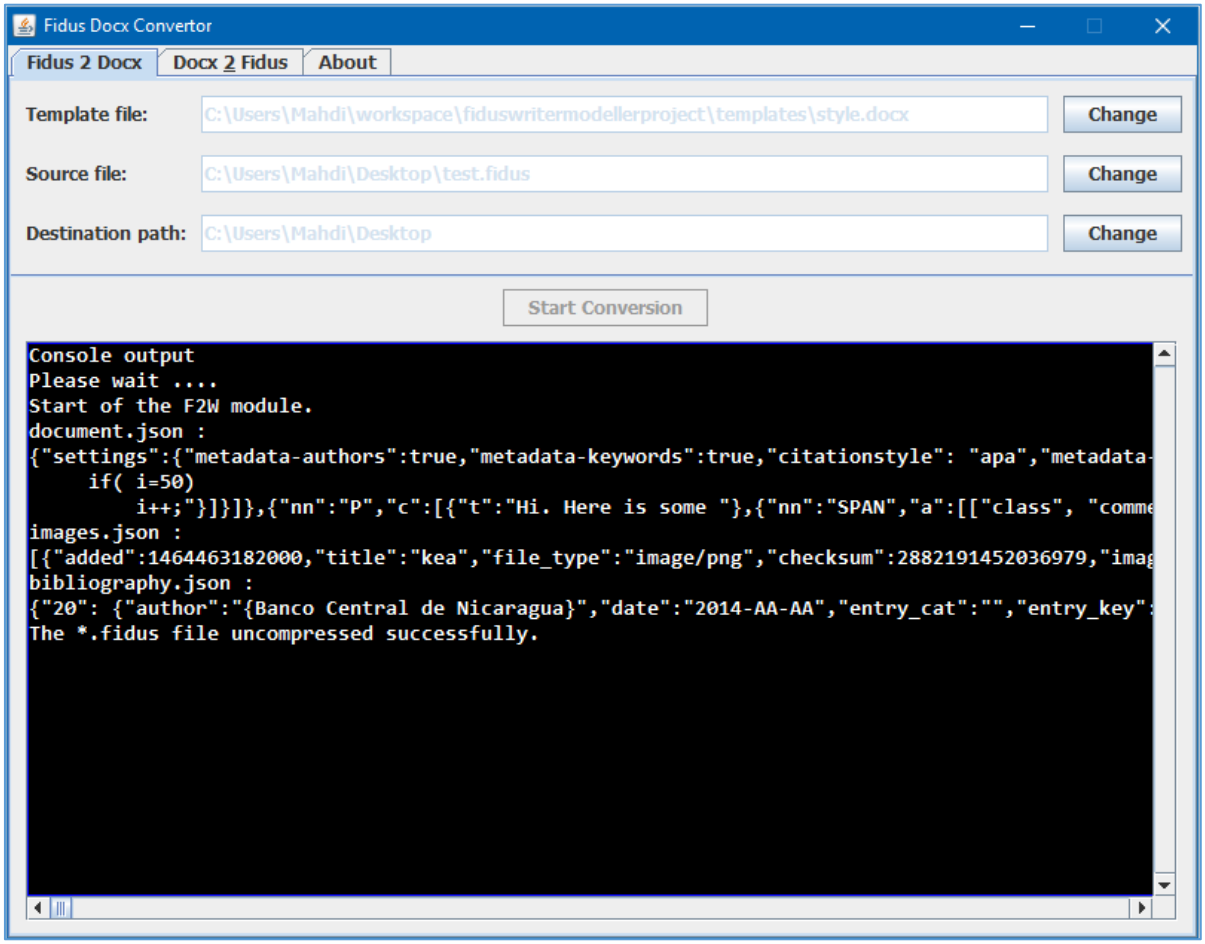
	python-docx 0.8.6[17]	docx4js[18]	html-docx-js[19]	mammoth.js[20]	Apache POI[21]	docx4j[22]
Language	Python	Node.js	JavaScript	JavaScript	Java	Java
Read, Write	RW	R	W	R	RW	RW
Title	×	×	×	×	×	✓
Subtitle	×	×	×	×	×	✓
Author	×	×	×	×	×	✓
Institute	×	×	×	×	×	✓
Abstract	×	×	×	×	×	✓
Keyword	×	×	×	×	×	✓
Date	×	×	×	×	×	✓
Header, Footer	×	✓	×	×	✓	✓
Footnote, Endnote	×	×	×	✓	✓	✓
Bookmark	×	✓	×	×	✓	✓
Cross-Reference	×	×	×	×	×	✓
Page number	×	×	×	×	✓	✓
Page break	✓	✓	✓	×	✓	✓
Hyperlink	✓	✓	✓	✓	✓	✓
Table	✓	✓	✓	✓	✓	✓
Figure	✓	✓	✓	✓	✓	✓
Equation	×	✓	×	×	×	✓
Caption	×	✓	×	✓	×	✓
Citation	×	×	×	×	×	✓
Bibliography	×	×	×	×	×	✓
Styling	✓	×	×	✓	✓	✓
Bold, Italic, Underlining	✓	✓	✓	✓	✓	✓
Superscript, Subscript	×	×	✓	✓	✓	✓
Bullet list	✓	✓	✓	✓	✓	✓
Numbered list	✓	✓	✓	✓	✓	✓
Heading	✓	✓	✓	✓	✓	✓
Comment	×	×	×	×	✓	✓
Extensibility without programming	×	×	×	×	×	✓
Total	9	12	9	11	15	28

[17] https://github.com/python-openxml/python-docx
[18] https://github.com/lalalic/docx4js
[19] https://github.com/evidenceprime/html-docx-js
[20] https://github.com/mwilliamson/mammoth.js
[21] https://poi.apache.org/
[22] http://www.docx4java.org/trac/docx4j

Fidus-Docx Converter Software

In this section, we explain some key technical features of the converter tool that made during this master thesis. The converter tool has a graphical user interface (GUI) with two different parts. Each part placed on a separate tab.

Figure 4-3: A view of the "Fidus Docx Converter" software



We tried to make the design of this tool as simple as possible. This tool includes a template file that contains some styles which help us to make the output file as similar as possible to the Fidus Writer format. If a user wants to change the look like of the output file, he or she must change those styles of the template file that their name starts with the term 'FW'. For example, 'FWTitle' is used for styling the title of the document, or 'FWChart' is used for all figures. The user can change the default address of the template file to some places that his or her customized file placed. When the user selects the source file and destination folder and then presses the 'Start Conversion' button, it starts to do the conversion and prints the information inside of the console panel.

The structure of the code has been divided into seven packages as follows:

1. ***auxiliary*:** This package contains the classes that have been used in different classes for some general staffs same as sorting, I/O and working with files (e.g. extracting or compressing the files).
2. ***mathEquations*:** The classes inside of this package are responsible for converting the math equations from OMML (Office Math Markup Language) which is used inside of Microsoft Office, to the LaTeX equations. LaTeX equation is the default formula inside of Fidus Writer. For doing this conversion we used the "fmath"[23] library, and two XSLT files from Microsoft (i.e. MML2OMML.XSL and OMML2MML.XSL). These two files are published within Microsoft Office. Currently, these two files are used in many libraries and software through the internet, however, the license of them has not been cleared from Microsoft. We called some people at Microsoft via email, however, they couldn't provide a clear answer about their licenses. We had a student version of Microsoft Office on our system, thus we used the files that had existed in our file system. However, we removed them from the public repository of the project. Therefore, if someone wants to use the converter must copy these two files from his or her file system into the 'templates/mathml_libs' folder of the project or download them via the internet at his own risk and placed them inside of that folder.
3. ***fidusWriter.fileStructure*:** This package and its classes are responsible for reading or writing the FIDUS files. When we want to convert a FIDUS file to the DOCX file we must extract the FIDUS file at first. This package manages this uncompressing and keeps the actual paths to the uncompressed files. Furthermore, when we want to convert a DOCX file to the FIDUS file, it is this package that moderates this compression.
4. ***fidusWriter.model*:** This package has three sub-packages. This package with its sub-packages has been used to model FIDUS file in JAVA. Each FIDUS file has three main files. They are *bibliography.json*, *document.json* and *images.json*. These three files are modeled inside of three sub-packages with the same name. Figure 4-4 illustrates the class diagram for the 'fidusWriter.model.documen' package.

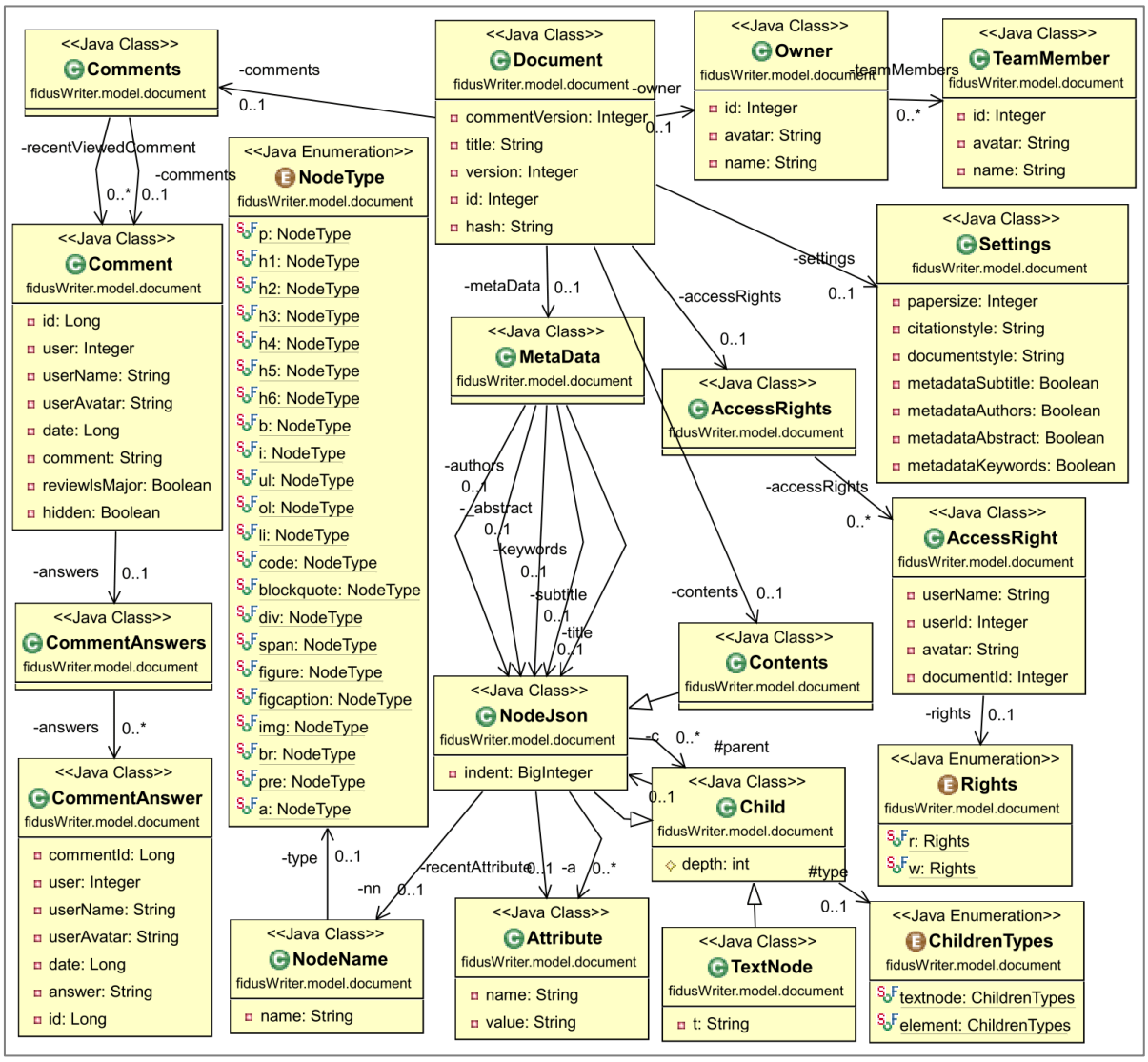
[23] More information about the "fmath" library can be found here: http://www.fmath.info/

5. ***fidusWriter.converter*:** This package has two sub-packages: ‘todocx’ and ‘tofidus’. As it is clear from their names the core codes that do the conversion and parse the input file and create the equivalent elements in the output file has been placed inside of this package.
6. ***window*:** This package has the code for moderating the GUI.
7. ***threads***: As this converter has been designed with the GUI, we designed some threads to prevent freezing of the GUI when the tool is in the middle of converting.

In the above section, we tried to give a top-down glance of the product. If more details related to the code is required, the comment inside of the source code files must be considered[24].

[24] The source code of the ”Fidus Docx Converter” project is accessible via the following public repository: https://github.com/mjza/MSThesis_Fidus_Docx_Converter

Figure 4-4: The class diagram of the 'fidusWriter.model.document' package



Challenges of the Conversion of Files to the Docx Format

When we make a program for converting a type of document files, to the DOCX format, the most challenge is to keep the formatting of the source file in the destination file. This can be done by providing a sample template file that contains some styles that have some properties similar to the features of the source file. For example, if the title in source file has the font Arial and font size of 20 points, we can have a similar style with the same settings. To provide this flexibility for the end users that they could change these formats by their own interests, it is possible to ask them to modify a copy of that template file.

Another challenge is to keep the metadata. For example, if the source file supports abbreviations while it does not support in MS Word, and we want to know which words are abbreviations in the destination file, it is possible to create a style inside of the template file

which is named ‘Abbreviation’ and assign that style to the abbreviations. In this way, if it is required converting the DOCX file back to its source format we will face with less data missing. It is not possible to extend the OOXML by some customized tags or properties. Furthermore, even if we could extend it, the solution is not a general one and does not mean that MS Word could support our customized tags. Thus the best way for extending MS Word is using the style facility.

Finally, the biggest issue is when we encountered with some incompatibility between features of DOCX and the source file. For example, in our Fidus-Docx converter, there was a considerable difference between the supported bibliography elements (i.e. sources) of Fidus Writer and MS Word. For example, MS Word supports the types like Film or Sound Recording while they are not supported by Fidus Writer and vice versa. Furthermore, even for those types that they were supported by the both software, there were some elements that were not supported for the equivalent types in both software. Thus we tried to inject the whole JSON object of the sources inside of the comment feature of the sources in DOCX file, and when we wanted to convert the file back to the FIDUS type we first create the source object based on the JSON inside of the comment if exists, and then try to modify it based on the equivalent elements that may be modified by the user inside of MS Word. However, even these kind of solutions cannot guarantee that data missing will not happen. It seems it would be a positive strategy that a new standard formed about the sources for the bibliography.

Challenges of the Conversion of the Docx Format to the Other Files Formats

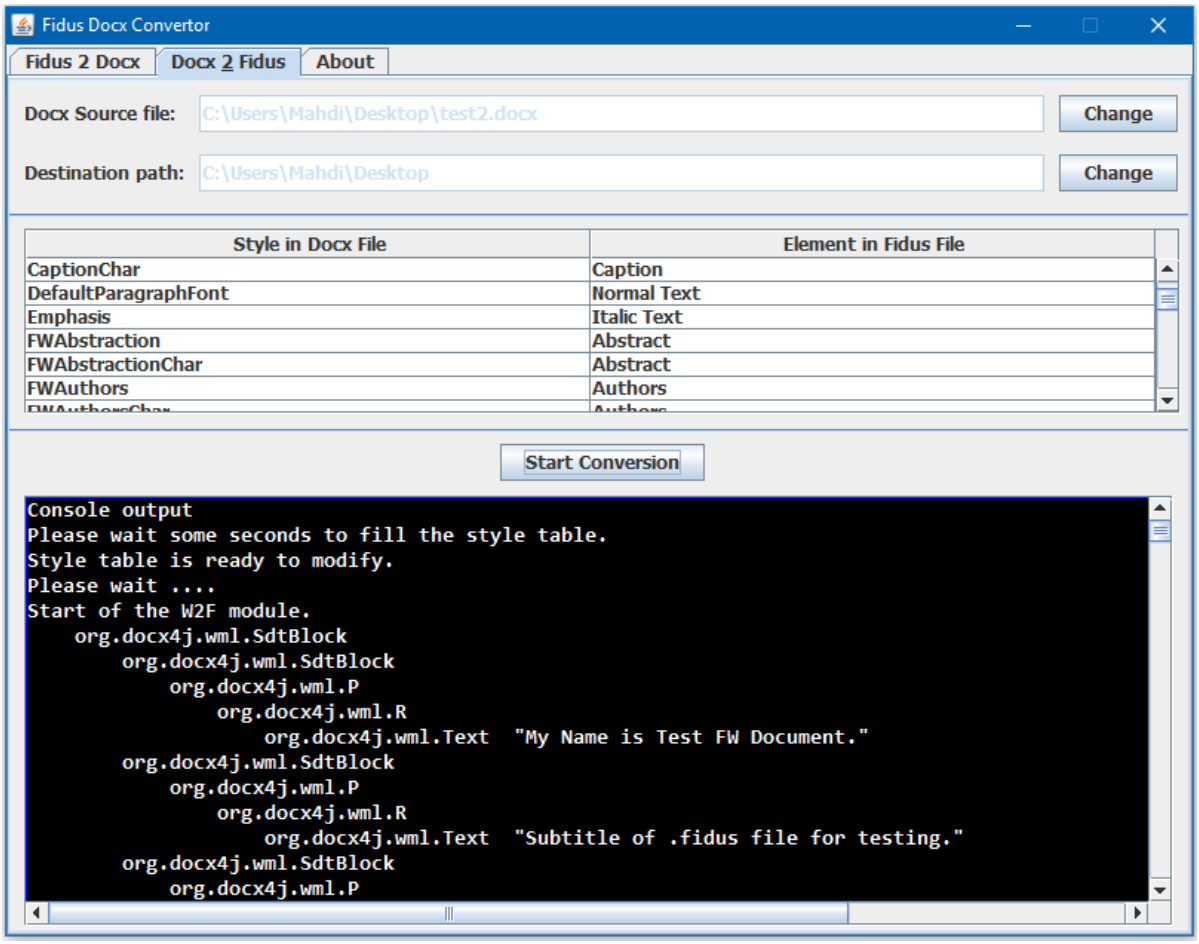
It seems reading the DOCX files is easier than creating it, however, it is completely a wrong assumption. MS Word has different versions and each of its versions has its own extensions. For example, if a shape is added to a DOCX file in MS Word 2013, it will add a shape with "word processing shape" standard which is not yet part of the OOXML standard. However, for the compatibility with older version of MS Word, a second version of the shape will be added to the document by using the VML drawing format which is defined clearly in OOXML standard. The third section of the OOXML standard, which is titled "*Markup compatibility and extensibility*" explain how a document is able to contain multiple versions of the same piece of content in different formats (Apache, 2016). These kinds of exceptions which are not limited, make the reading of the DOCX files more difficult than creating a DOCX file.

Another issue is how to transfer metadata to the destination format. For example, during our investigation on the DOCX source files of some scientific papers, we noticed that in most

cases, scientists didn't add the title or the abstraction in a correct way. The best cases between considered papers were those papers that the authors assigned a suitable style to the elements. Furthermore, we saw in some cases, people used some template files that they had different style names. For example, in one paper they used 'H1' instead of 'Heading 1'. In these cases, it is required getting the help of end users and requesting him or her to adjust a mapping table for the used styles inside of the DOCX file. This is the way that we designed the 'Docx 2 Fidus' section of our converter. When the user selects the source file, a list of used styles in the DOCX file is illustrated on the left side of the mapping table and the user can select the equivalent element in FIDUS format (Figure 4-5).

Another problem here is unsupported features in the destination format as well. This problem could cause unavoidable data missing. For example, tables are not supported in Fidus Writer yet, or some types of the sources of the bibliography or cross reference or even endnotes. And almost all of the web-based WYSIWYG authoring tools cannot support multi-column page layout.

Figure 4-5: A view of the 'Docx 2 Fidus' tab in the Fidus Docx Converter



Chapter 5- Future Works

A. Ideas for supporting authoring and reviewing tools in general

For supporting web-based and desktop-based collaborative authoring and reviewing tools generally, there are some limitations and shortages which can be covered by future attempts:

A.1. Requirements for developing new abstracts and standards

1. Defining a new standard for document files based on JSON, which fulfills requirements of both of the desktop-based and web-based authoring environments. JSON format is suitable for both desktop-based and web-based applications, especially for web-based ones. It seems the future software applications trend towards web-based environments. Such kind of standard can make a revolution in relation to making these two types of environments as closer as possible to each other. An important shortage in almost all of the current standards like OOXML or ODF is overlooking an efficient way of the tracking of changes. Tracking of changes can be implemented efficiently by a database structure. Thus, during the designing of such standard not only it has to have a model of the document in JSON format, but also a simple database file (e.g. SQLite file) inside of the compressed file can be used for recording the evolution of the document and records the changes. This new standard must not be developed without considering current existing standards for a document (e.g. OOXML or ODF). This new standard must be provided without any kind of limitations otherwise again a story like creating OOXML despite the existence of ODF could be happened by the private companies and then it cannot reach its goal.
2. Creating a network protocol for making the real-time collaborative possibility between desktop-based and web-based authoring and reviewing tools of different companies, can be the best solution for easier collaboration between different authoring tools.
3. Designing an abstract architecture for publishing, reviewing, editing and revision stages of a document are something that is required strongly for supporting the versioning concept in authoring and reviewing tools.

A.2. Requirements for improvement of the current existing standards and software tools

1. One of the most important incompatibilities between different document file formats is the bibliography section. We observed different standards has different fields for each type of resource. For example, in Fidus Writer it is possible to define five different identification numbers (ISSN, ISBN, ISRN, DOI and EID) for a book, however, in MS Word it is just possible to define two identification numbers for a book. Or in MS Word there are some new types of resources like 'film', 'CD', 'Sound Recording' which are required for the digital world, while they are not existing in some other authoring applications like Fidus Writer. Thus, it seems to define a standard for bibliography can be useful for compatibility between different file formats at the time of conversion between different standards.
2. Improving existing standards or defining a new standard for the drawing of shapes in the documents which is compatible with both desktop-based and web-based environment is another requirement. Desktop-based authoring tools have more capabilities for supporting complicated structures, however, web-based tools are limited by the capabilities of browsers. Thus, such kind of standard must focus on modeling some structures like SVG – which is supported in almost all of the major browsers – but in JSON format. Recently Microsoft has introduced a new standard for drawing shapes in MS Word 2013 and later which is called "word processing shape".
3. Currently almost all of the web-based and desktop-based authoring tools suffer the lack of a powerful citation tool that could connect to a comprehensive database which contains bibliographic data of latest academic books and papers. Such a tool can be presented as an add-on for different authoring tools and get the help of some considerable database like Google Scholar. The tools must be able to import the citation and the resource in a correct format that is accepted by the targeted authoring application and not a simple text. Importing a correct citation as a simple text can be useful, however, it is not the desired goal.
4. Moreover, developing some add-ons that evaluates the existing citations and sources in a document and makes some warnings about mistakes related to details of each citation can be useful for scientific purposes and especially for reviewers as well.

A.3. Requirements for improvement of the existing authoring and reviewing software tools

1. Supporting for the workflow "enhancing the commenting facility of MS Word with semantics (i.e. richer structures), and preserving such rich comments after a conversion to other formats" can lead to useful products. Here are two examples of semantic comments:
    - "The selected text is related to my topic of interest <topic...>"
    - "The selected text is related to something on the internet with this URL"

   Feature request can be the ability for filtering such comments, e.g. "display only those comments that are related to <topic...>". Therefore, it is required defining how semantic structures in comments can be preserved in other formats. Furthermore, it could be a useful tool to develop some add-ons in MS Word to support rich comments and operating on them.
2. Pleasereview is a valuable product for reviewing of DOCX documents however, it has three major problems. The features of the Pleasereview can be developed for other document format files and other authoring tools if the following problems resolved:
    a. First of all, it is not a free product and even does not have a demo version. Thus, it is not surprising that just 8% of our interviewees know this product or preferred to use it. Therefore, it could be valuable if a free version of this product provided for scientific communities at least.
    b. Pleasereview application is a web-based application, however, its design is a classical design which is not counted as a user-friendly design nowadays. It can be helpful if a better flat and modern design suggested for this kind of product.
    c. In the result of our online survey, we discussed that users prefer to have both authoring and reviewing tools as an integrated application. Thus, it is not a positive experience for the end users of Pleasereview that they cannot edit the document directly and they require entering their edits in a separate textbox. Therefore, making some add-ons that bring the ideas behind of the Pleasereview application inside of the current existing authoring tools can be a more successful idea rather than developing separated reviewing tools.
3. A desirable idea for scientific societies is to provide a kind of public web-based reviewing social network that people from different part of the world could review

each other's scientific writings. This kind of websites must consider a powerful legal mechanism to prevent plagiarism.

4. Currently, the major desktop-based authoring applications support multi-column page layout. Multi-column page layout is used as the default layout in many scientific journals. Unfortunately, none of the web-based authoring tools can support multi-column page layout. It is required testing different ideas for managing a reach textbox area with the help of JavaScript and find the best solutions for resolving this lack.
5. Finally, providing some algorithm for developing the Git technologies for supporting offline collaboration on complex document files like DOCX files or FIDUS files can be a worthwhile attempt.

B. Ideas for improving "Fidus Docx Converter" software

Related to "Fidus Docx Converter" the following improvements can be done:

1. Currently, Fidus Docx Converter is a desktop-based application. As Fidus Writer is a web-based application and this converter was aimed to be used inside of the OSCOSS project, thus, it is required extracting the code from this project, patching it as a server-side CGI, and developing a web-based GUI for that. (While it is possible to use this project as a Java Applet in client side, however, because Java Applet is not supported in all browsers anymore, this method is not suggested). Currently, just one package of the code is related to GUI.
2. At the moment there is no powerful JavaScript library for reading and writing DOCX files efficiently (There are some libraries, however, suffers many limitations). Developing a JavaScript library for working with DOCX files is not only a positive option for development of a converter for Fidus Writer in client-side but also it can be used in other web-based authoring tools.
3. Developing individual libraries in Java and C# for creating Fidus files programmatically can be a positive option for the future spreading of the Fidus Writer. However, for creating such kind of file the first requirement is defining a standard for the FIDUS format file.
4. Fidus Docx Converter uses two XSLT files from Microsoft for converting math equations from OMML standard to MML standard. Then the math equations from MML format are transferred to LaTeX standard by the help of "fmath" library. As we didn't receive any clear answer from Microsoft related to the licenses of those

two XSLT files, we didn't publish them inside of the public repository on GitHub. Thus, it is required to finding a new solution for converting math equations back and forth between OMML and LaTeX standards.

5. The converter ignores the tables as Fidus Writer does not have any facility for supporting tables. It is possible to make a picture out of a table in a DOCX file and insert that picture in FIDUS file for now. Another possibility is that define a structure for tables inside the FIDUS file format standard and develop the code for forming such kind of equivalent tables inside the targeted FIDUS file. Furthermore, reading of the table from the future standard of the FIDUS file and import it in the targeted DOCX file is required in the future. Cross-references and end-notes are another features that we didn't cover them in our converter as they are not supported in Fidus Writer.
6. Our converter just reads one type of the pictures which is inline pictures. It is required developing the code to support other types of the pictures inside of the DOCX files.
7. Currently, our converter ignores the citations that are simple text in both FIDUS and DOCX files and import them as simple text in the destination format. Developing the code in such a way that extracts the citations based on some predefined regular expressions can be useful. Such kind of improvement can be worthwhile as it helps to import citations and sources in their correct format in the destination file.
8. We saw in most of the scientific DOCX files people insert cover page elements like the title as simple texts as well. Thus, it would be a positive attempt to detect them in DOCX files with some intelligent methods or by using regular expressions.
9. Supporting for the workflow "change a DOCX document with *track changes* option enabled, and merge these changes into a FIDUS version of the original Word document" is another possibility for improvement of the Fidus Writer and the converter. To make such improvement, it is required:
    a. Discovering how tracked changes are represented in the DOCX files.
    b. Discovering how to retrieve these data from the DOCX files via the docx4j library.
    c. Defining a new structure in FIDUS file format standard that is able to represent the track changes (As currently, Fidus Writer does not support it).

Chapter 6 - Conclusion

Collaboration on authoring and reviewing of a scientific paper is not a new phenomenon, however, it is at a turning point in its lifetime and the form of the collaboration is changing. Nowadays, by appearing web-based authoring and reviewing tools which contain collaborative ability, the acceleration towards web-based collaborative authoring tools is increasing. Web-based software applications have some pros and cons. For example, while updating the application is easier for developers, however, connecting to the internet is a requirement for using a web-based application. Thus, the best solution for preventing developing two versions of an application (i.e. desktop-based and web-based) is developing an application as a browser add-on application. For example, rather than developing a pure web-based application, it is better to develop an application which is installed on browsers like Google Chrome or Mozilla Firefox. Such kinds of browser-based applications are able to be used online or offline.

Offline collaboration was a forgotten requirement at the time of the developing document file format standards like OOXML and WYSIWYG authoring desktop-based application. Although web-based authoring and reviewing tools provided an acceptable real-time collaborative environment, however, offline collaboration with some features that GitHub provides like branching and merging is something that has not been resolved and requires more works.

Providing some new standards for document file formats based on the JSON format which is more suitable for the web-based applications rather than XML format, and benefiting from simple database structures like SQLite for recording the changes of a document file can make a positive progress in the evolution of authoring and reviewing tools. This JSON structure can be stored even completely inside of the database file. Thus, it is required individually studying the advantages and disadvantages of such kind of new structure for document files.

Reaching some common standards for different usual features of the document files is required. It can be seen the incompatibilities between some features of the document file formats (e.g. differences between bibliography part of DOCX and FIDUS files) can lead to data missing at the time of conversion between document files.

Missing of formatting is the most reported complaints that users experienced at the time of converting document files. Thus, one of the key features for successfulness of a converter is the consideration of keeping the formats of different parts of the document.

Developing some add-ons for authoring tools that collect the most important features that are used in scientific documents in one menu or one tab, can help scientists to use their authoring tools in a better way. We observed that more than 85% of the scientists do not know MS Word have some features for inserting the title of the document in a suitable textbox. Thus, if an add-on collects these kinds of features in a tab in MS Word ribbon can help the people who use this tool for scientific purposes to find common features easier. Even software companies which provide authoring tools can present a scientific mode in their tools that make it more convenient for scientific usages.

Bibliography

Ballesteros, S., & Bogic, T. (2016, April 26). *DOCX STANDARD SCIENTIFIC STYLE*. Retrieved from science.ai Inteligent Science Publishing: http://scienceai.github.io/docx-standard-scientific-style/

A Short History of Git. (2016, June 15). Retrieved from git: https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git

A.H. (2013, May 30). *Xtext - multiple files language.* Retrieved from stackoverflow: http://stackoverflow.com/questions/16831139/xtext-multiple-files-language

Apache. (2016, June 27). *OOXML/Markup Compatibility and Extensibility.* Retrieved from Apache OpenOffice Wiki: https://wiki.openoffice.org/wiki/OOXML/Markup_Compatibility_and_Extensibility

Coffey, J., McGoldrick, N., McGoldrick, M., & McGoldrick, J. (2000). *Marriage of Minds: Collaborative Fiction Writing.* National BOOK NETWORK.

Definition of WYSIWYG in English. (2016, July 7). Retrieved from Oxford Dictionaries: http://www.oxforddictionaries.com/definition/english/wysiwyg

EISGroup. (2015, October 15). *OSCOSS: A shared platform for Opening Scholarly Communication in the Social Sciences.* Retrieved from Enterperise Information Systems: http://eis.iai.uni-bonn.de/Projects/OSCOSS.html

Find and replace text by using regular expressions. (2016, June 15). Retrieved from Microsoft Office Support: https://support.office.com/en-us/article/Find-and-replace-text-by-using-regular-expressions-Advanced-eeaa03b0-e9f3-4921-b1e8-85b0ad1c427f?ui=en-US&rs=en-US&ad=US&fromAR=1

Instant Update. (2016, June 21). Retrieved from Scribd: https://www.scribd.com/doc/5447595/Instant-Update

ISO. (2008, April 2). *ISO/IEC DIS 29500 receives necessary votes for approval as an International Standard.* Retrieved from ISO: http://www.iso.org/iso/news.htm?refid=Ref1123

Lomas, N. (2014, September 22). *Authorea Nabs $610k For Its Bid To Become A 'Google Docs For Scientists'*. Retrieved from TechCrunch: https://techcrunch.com/2014/09/22/authorea-seed/

LOPEZ, N. (2015, October 10). *Microsoft Word 2016 is getting collaborative editing for OneDrive files.* Retrieved from TNW Brand Boost: http://thenextweb.com/microsoft/2015/08/13/microsoft-word-2016-is-getting-collaborative-editing-for-onedrive-files/#gref

Lowry, P. B., Curtis, A., & Lowry, M. R. (2004). Building a taxonomy and nomenclature of collaborative writing to improve interdisciplinary research and practice. *Journal of Business Communication, 41.1*, 66-99.

Oracle. (2016, June 15). *Introduction to JAXB.* Retrieved from Oracle Java Documentation: https://docs.oracle.com/javase/tutorial/jaxb/intro/

Pant, T. (2015, Februery 18). *Online, Collaborative Editing with Etherpad.* Retrieved from sitepoint: https://www.sitepoint.com/online-collaborative-editing-etherpad/

Peat, J., Elliott, E., Baur, L., & Keena, V. (2002). *Scientific writing: easy when you know how.* BMJ Books.

PleaseReview™ for Document Collaboration. (2016, June 15). Retrieved from amplio: http://www.amplioservices.com/products/pleasereview

PleaseTechLtd. (2014, April 16). *PleaseReview, document review and co-authoring software in action.* Retrieved from YouTube: https://www.youtube.com/watch?v=pYo5F3Fpymo

Plutext. (2016, May 11). *Docx4j Helper Word AddIn.* Retrieved from docx4j Blog: http://www.docx4java.org/blog/2016/05/docx4j-helper-word-addin-new-version-v3-3-0/

Preservation, D. (2016, June 22). *DOCX Strict (Office Open XML), ISO 29500-1: 2008-2012.* Retrieved from Sustainability of Digital Formats Planning for Library of Congress Collections: http://www.digitalpreservation.gov/formats/fdd/fdd000400.shtml

Spier, R. (2002). The history of the peer-review process. *TRENDS in Biotechnology*, 357-358.

Standard ECMA-376. (2016, June 22). Retrieved from ECMA: http://www.ecma-international.org/publications/standards/Ecma-376.htm

TechNet. (2011, August 05). *Overview of the XML file formats in Office 2010.* Retrieved from Microsoft Technet: https://technet.microsoft.com/en-us/library/cc179190.aspx

TechNet. (2011, August 05). *Overview of the XML file formats in Office 2010.* Retrieved from Microsoft TechNet: https://technet.microsoft.com/en-us/library/cc179190.aspx

TechNet. (2014, March 14). *XML file name extension reference for Office 2013.* Retrieved from Microsoft TechNet: https://technet.microsoft.com/en-us/library/cc179191(v=office.15).aspx

Wilm, J. (2016, June 23). *The idea behind the FidusWriter.* Retrieved from Fidus Writer: https://www.fiduswriter.org/how-it-works/

Appendix A - Distribution of Participants in Our Online Survey

Table 6-1: Number of the participants in our online survey per university

Institution	Country	#	Institution	Country	#
University of Macquarie	Australia	13	University of Potsdam	Germany	4
University of New England	Australia	1	University of Bombay	India	1
University of South Wales	Australia	1	University of Delhi	India	3
University of Swinburne	Australia	1	University of Isfahan	Iran	3
University of Salzburg	Austria	2	University of Kermanshah	Iran	1
University of Vienna	Austria	3	University of Mashhad	Iran	13
University of Carleton	Canada	1	University of Tehran	Iran	9
University of Macau	China	3	university of Kurdistan	Iraq	2
University of Aarhus	Denmark	1	University of Tokyo	Japan	1
University of Copenhagen	Denmark	1	University of Netherlands	Netherlands	1
University of Paris	France	1	University of Malaga	Spain	1
GESIS Institution	Germany	11	University of Murcia	Spain	2
University of Aachen	Germany	2	University of Pompeu	Spain	3
University of Berlin	Germany	4	University of Cambridge	UK	2
University of Bochum	Germany	1	University of Cardiff	UK	3
University of Bonn	Germany	31	University of Kent	UK	1
University of Bremen	Germany	1	University of Leeds	UK	1
University of Cologne	Germany	3	University of London	UK	2
University of Dortmund	Germany	1	University of Manchester	UK	4
University of Frankfurt	Germany	1	University of Sunderland	UK	1
University of Gottingen	Germany	1	University of Warwick	UK	1
University of Hagen	Germany	1	University of Fayetteville	USA	2
University of Hamburg	Germany	12	University of Florida	USA	3
University of Karlsruhe	Germany	4	University of Grambling	USA	1
University of Konstanz	Germany	1	University of Harvard	USA	1
University of Kassel	Germany	1	University of Maryland	USA	3
University of Leibniz	Germany	1	University of Pennsylvania	USA	1
University of Munster	Germany	1	University of Syracuse	USA	9
University of Osnabruck	Germany	1	University of Washington	USA	2
University of Passau	Germany	1	Not Specified	Unknown	24
Total	**###**	**108**	**Total**	**###**	**105**
Final Number of Participants					**213**

Figure 6-1: Distribution of the participants in our online survey in a world map

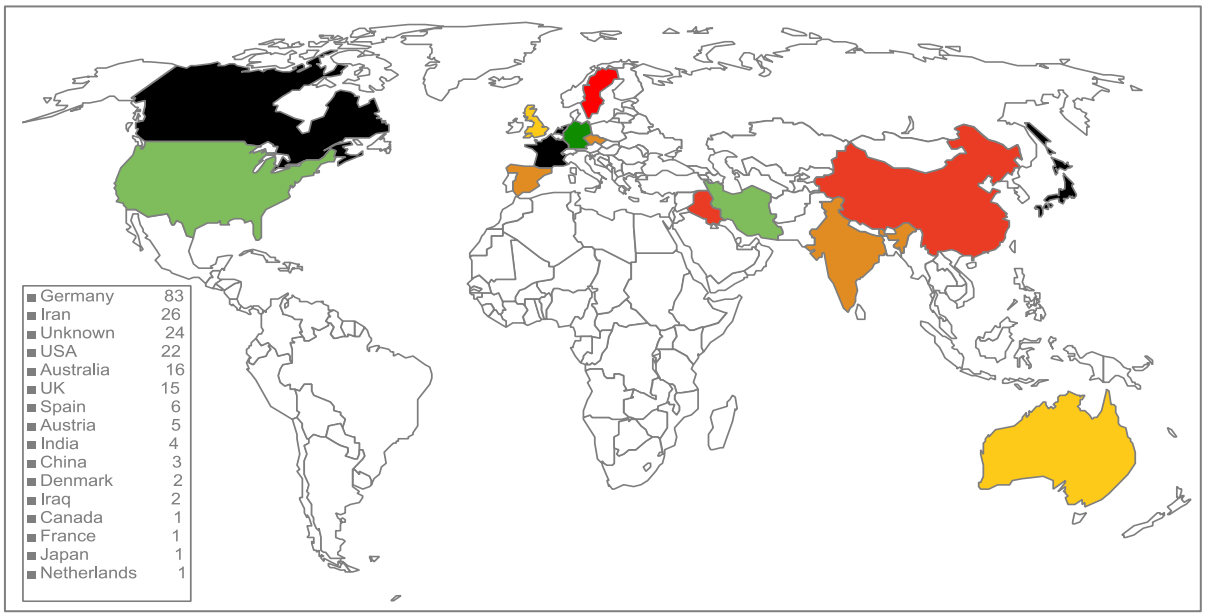
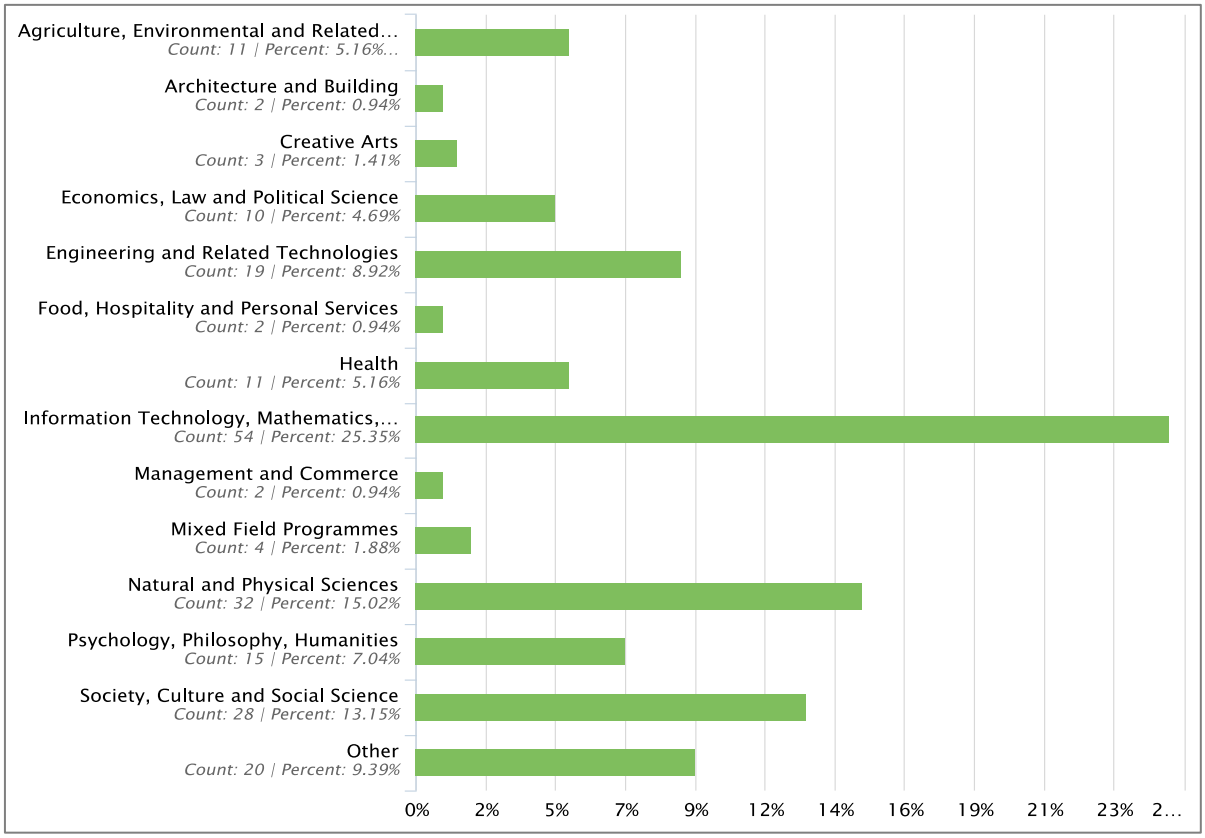


Figure 6-2: Distribution of the field of study of the participants in our online survey.



Appendix B - Detailed Result of the Survey on the DOCX Files

Table 6-2 : Result of the survey on the 27 real scientific paper's source files[25]

Document Name	Feature name→		Title	Subtitle	Author	Institution	Abstraction	Keywords	Date	Headers/Footers	Footnotes/Endnote	Bookmarks	Cross-References	Page numbers	Hyperlinks	Tables	Figures	Equations	Captions	Citations	Bibliographies	Stylings	Indexes	Bullet lists	Numbered lists	Headings	Abbreviations
D0		U[26]	1	1	1	1	1	0	0	0	0	0	0	0	3	0	0	0	9	114	1	0	0	0	15	17	105
		C	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	105
D1		U	1	1	1	3	1	0	0	1	1	0	0	1	8	7	1	0	7	64	1	0	0	0	6	6	217
		C	0	0	0	0	0	0	0	1	1	0	0	1	0	7	1	0	0	0	0	0	0	0	5	5	217
D2		U	1	1	1	1	1	1	0	1	1	0	0	1	0	0	3	0	3	37	1	0	0	0	0	7	150
		C	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	150
D3		U	1	1	1	1	1	0	0	0	14	0	0	1	35	4	4	0	8	68	1	0	0	0	17	19	195
		C	0	0	0	0	0	0	0	0	14	0	0	1	35	4	4	0	0	68	1	0	0	0	17	19	195
D4		U	1	1	1	1	1	1	0	0	1	2	0	1	3	6	0	0	5	8	1	0	0	0	6	8	134
		C	0	0	0	0	0	0	0	0	1	2	0	1	3	6	0	0	0	0	0	0	0	0	0	0	134
D5		U	1	1	1	1	1	1	0	0	23	0	0	1	2	10	5	4	13	100	1	0	0	0	13	5	139
		C	0	0	0	0	0	0	0	0	23	0	0	1	2	10	5	0	0	0	0	0	0	0	0	0	139
D6		U	1	0	1	1	1	1	0	1	27	0	0	1	4	12	0	10	15	53	1	0	0	0	0	21	200
		C	0	0	0	0	0	0	0	1	27	0	0	1	4	12	0	0	8	0	0	0	0	0	0	0	200
D7		U	1	1	1	1	1	1	0	0	16	0	0	1	9	0	1	0	1	51	1	5	0	0	21	14	228
		C	0	0	0	0	0	0	0	0	16	0	0	1	0	0	0	0	0	0	0	3	0	0	21	14	228
D8		U	1	0	0	0	1	0	0	0	23	0	0	1	2	4	5	0	9	19	1	9	2	0	11	13	132
		C	0	0	0	0	0	0	0	0	23	0	0	1	2	4	5	0	0	0	0	5	2	0	11	13	132
D9		U	1	1	1	1	1	0	0	1	0	0	0	1	2	4	1	0	5	56	1	0	0	0	11	12	53
		C	0	0	0	0	0	0	0	1	0	0	0	1	0	4	1	0	0	0	0	0	0	0	0	0	53
D10		U	1	0	1	1	1	1	0	0	15	0	0	1	3	7	5	0	12	112	1	5	0	0	11	12	130
		C	1	0	0	0	0	0	0	0	15	0	0	1	3	7	2	0	0	112	1	3	0	0	0	12	130
D11		U	1	1	0	0	1	0	0	0	10	0	0	1	2	0	0	0	0	36	1	0	0	0	7	8	24
		C	1	0	1	0	0	0	0	0	10	0	0	1	2	0	0	0	0	0	1	0	0	0	7	8	24

[25] This table continues in the next page.

[26] In Table 6-2*U* means '*Number of Usage*', and *C* means '*Number of Correct Usage*'.

Abbreviations	47	47	122	122	141	141	39	39	41	41	17	17	31	31	184	184	93	93	134	134	68	68	67	67	44	44	159	159	132	132	3026	3026
Headings	8	0	21	0	6	0	10	0	9	0	8	0	10	0	31	31	14	14	10	10	7	7	13	13	23	0	0	0	14	13	326	152
Numbered lists	6	6	16	0	13	10	0	0	1	0	1	0	1	0	8	8	14	14	18	18	10	9	20	13	25	25	4	4	0	0	255	168
Bullet lists	3	3	3	3	1	1	0	0	0	0	0	0	0	0	5	5	6	6	3	3	8	8	0	0	0	0	0	0	5	5	42	42
Indexes	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Stylings	0	0	6	4	0	0	0	0	0	0	0	0	0	0	6	5	10	7	8	6	4	3	7	7	50	50	30	30	6	6	152	135
Bibliographies	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	0	1	0	1	0	27	10
Citations	16	0	30	0	53	0	14	0	14	0	14	0	16	0	39	0	4	0	10	10	28	0	36	36	70	0	150	0	20	0	1232	226
Captions	6	2	11	5	7	0	3	0	3	0	2	0	0	0	3	0	6	0	6	6	0	0	7	7	40	8	8	5	3	3	192	44
Equations	16	13	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	29	29	0	0	0	0	71	42
Figures	4	3	8	8	1	1	0	0	0	0	0	0	0	0	6	6	5	5	5	5	0	0	0	0	8	8	5	5	3	3	70	62
Tables	2	1	3	3	10	10	1	0	1	0	0	0	0	0	0	0	1	1	1	1	0	0	7	7	3	3	3	3	0	0	86	83
Hyperlinks	1	0	2	1	1	0	0	0	0	0	0	0	0	0	33	0	19	0	11	10	3	2	3	2	1	1	1	0	8	8	156	78
Page numbers	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	0	0	1	1	1	1	1	1	1	1	23	22
Cross-References	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	24	0	0	6	6	0	0	0	0	0	0	0	0	0	0	30	30
Bookmarks	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	66	66	0	0	0	0	0	0	70	70
Footnotes/Endnotes	2	0	42	42	0	0	0	0	0	0	0	0	0	0	1	1	18	18	16	16	0	0	0	0	0	0	1	0	0	0	211	208
Headers/Footers	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	1	1	1	2	2	2	2	1	1	14	14
Date	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Keywords	1	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	1	0	0	0	0	0	0	1	0	0	0	1	0	13	1
Abstraction	1	0	1	0	0	0	1	0	0	0	1	0	1	0	1	0	1	1	1	0	1	1	1	1	1	0	1	1	1	1	25	5
Institution	1	0	0	0	4	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	0	1	1	1	0	2	2	0	0	28	5
Authors	1	0	1	0	1	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	0	25	5
Subtitle	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	12	1
Title	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	27	8
Feature name↑	**U**	**C**	**U**	**C**	**U**	**C**	**U**	**C**	**U**	**C**	**U**	**C**	**U**	**C**	**U**	**C**	**U**	**C**	**U**	**C**	**U**	**C**	**U**	**C**	**U**	**C**	**U**	**C**	**U**	**C**	**U**	**C**
Document Name	**D12**		**D13**		**D14**		**D15**		**D16**		**D17**		**D18**		**D19**		**D20**		**D21**		**D22**		**D23**		**D24**		**D25**		**D26**		**Total**	

We have to explain how this data was collected and what criteria were used for counting a sample of a feature as a correct used or a wrong used one. Here is the explanation of the criteria for critical features. Some simple features like headers and footers were ignored.

1. The first group of features includes title, subtitle, authors, institution, abstraction, keywords and date. Usually, if they exist in a document they appear one time on the first page of the document which is called cover page. If the item was created by using *INSERT>Explore Quick Parts>Document Property>...* menu (Figure 6-3) or a suitable style was assigned to the item (name of the style was matter, for example, the Title style is suitable for the document's title) then that item counted as a corrected made one as well.

Figure 6-3 : The correct way of inserting the title in a DOCX file

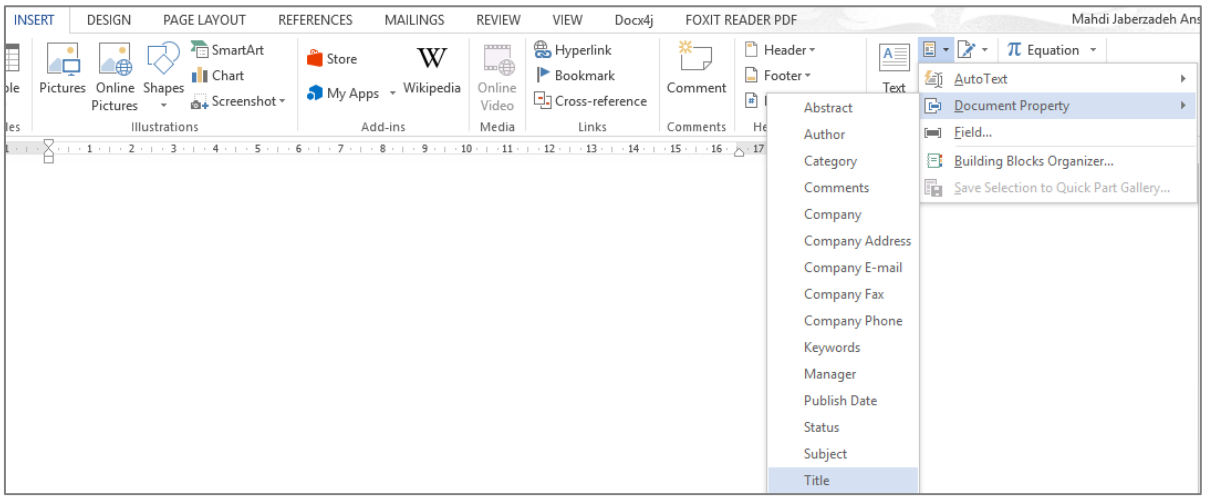
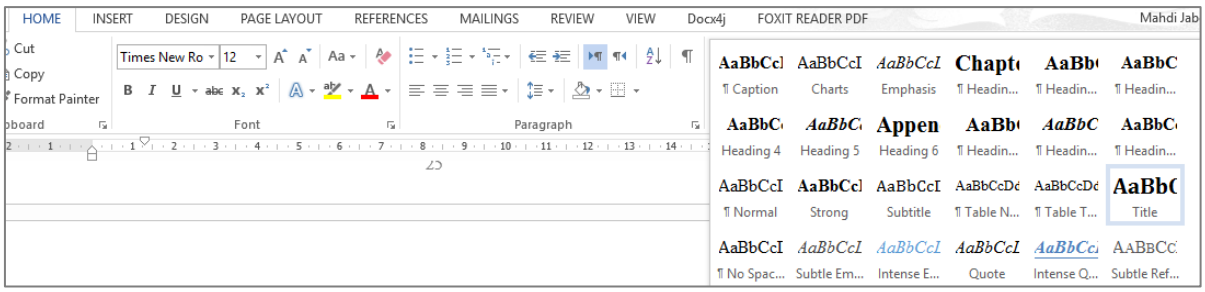


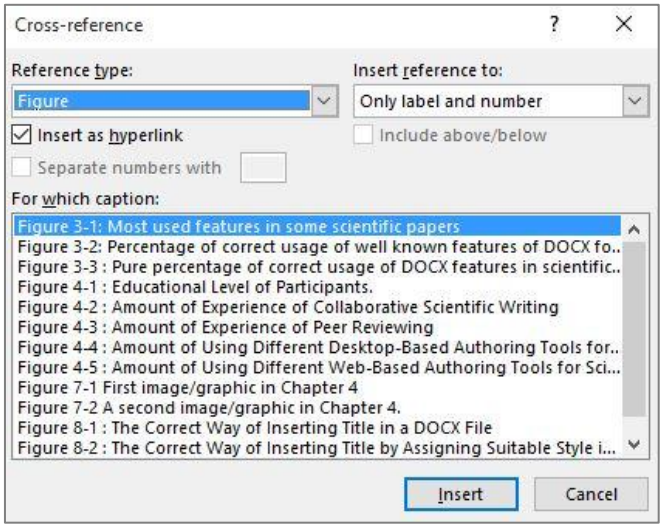
Figure 6-4 : The correct way of inserting the title by assigning suitable style in Microsoft Word



2. The next group of features contains number lists, footnotes, endnotes, bookmarks, headings, and captions. We have to count them line by line and manually. However, when we want to count the number of the correct usages of them it is easy. We can use '*INSERT > Cross-reference*' menu. This menu opens a window (Figure 6-5)

that lists all cases of each of these features. We used this option to count the number of the correct captions or the correct headers and so on.

Figure 6-5 : The cross-reference window in MS Word



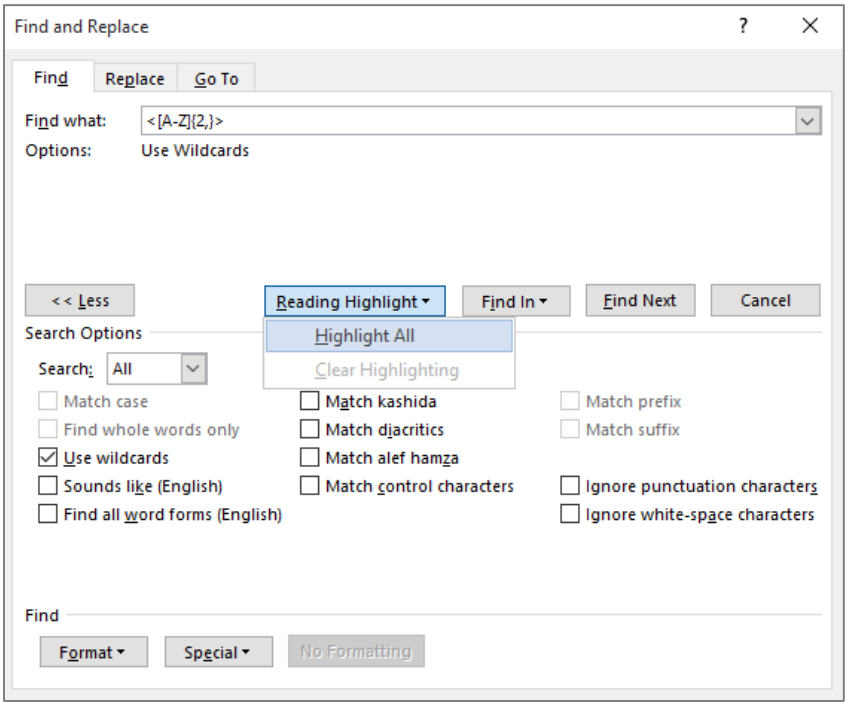
3. The third feature is the hyperlink. To count the number of the hyperlinks we can easily search for '*http://*' or '*https://*' terms in the context by using '*Advanced Find*' facility of Microsoft Word. It is possible to press the '*Highlight All*' key to highlight all the hyperlinks and it illustrates the number of highlighted items. For counting the number of the correct hyperlinks, we had to press 'Alt+F9', then MS Word reveals field codes related to features. We can search for the term '`^d HYPERLINK`' and using '*Highlight All*' again to count the number of the correct hyperlinks this time.

   Once again, what is the benefit of using '*INSERT>Hyperlink*' menu when we want to make a hyperlink rather than write the hyperlink as a simple text? In fact, when we convert the file to PDF format for publishing, the readers can easily click on the hyperlinks and redirect to the links easily. Furthermore, search engines and other programs who want to evaluate the content of the file, they don't require running regular expression algorithms on the document to extract its hyperlinks for further evaluations. Usually, regular expression algorithms are time-consuming algorithms.

4. For finding the number of the abbreviations, we used a simple regular expression. It is easy to use the regular expressions (Find and replace text by using regular expressions, 2016) in MS Word to find some samples of a pattern. It just requires

checking the "*Use wildcards*" option as is illustrated in Figure 6-6. For abbreviations we assumed every word with capital letters which has two or more letters, is an abbreviation. There are some exceptions, thus the extracted number maybe includes some small tolerance. The regular expression that we used was '`<[A-Z]{2,}>`'.

Figure 6-6 : Using the regular expressions in "Advanced find window" of MS Word



5. When it turns to finding the number of the citations, the story is more complicated. We have to use regular expression again. However, the problem appears when different standards are used in the documents for citations or some nonstandard ways like writing citation as a simple text have been used. We tested more than four different regular expressions, however, finally we selected '`\([!\)]@[0-9]{4}\)`' because it can cover most of the cases. However, this expression can find those citations that follow APA[27] or other standards. For example, if the author used square brackets instead of parenthesis, then it can cause no result for searching the text with this pattern. For finding the number of the correct citations, we have to use 'Alt+F9' to see field codes again and use '`^d ADDIN EN.CITE`' term in the advanced search.

[27] More information about the APA style can be found here: http://www.apastyle.org/

6. For finding the number of the captions we can use a regular expression again. For example, if figures have captions in the form of ‘Figure 1’, then the regular expression that can be used to find all the captions of this kind can be something like ‘`Figure [0-9]@:`’.

These six methods were the most important techniques that we used to count the number of different used features in sample DOCX files.