Automatic Extraction of Formal Features from Word, Excel, and PowerPoint Productions in a Diagnostic-Assessment Perspective

Jean-Philippe Pellet
jean-philippe.pellet@hepl.ch

Morgane Chevalier
morgane.chevalier@hepl.ch

Teaching & Research Unit for Media & ICT
Lausanne University of Teacher Education (HEP-VD)
Lausanne, VD, Switzerland

***Abstract*—The large-scale evaluation of documents produced by hundreds or thousands of students poses a challenge in terms of human and time resources and equity of treatment. In this paper, we discuss the development and use of methods for automatic extraction of formal properties of Microsoft Word, Excel, and PowerPoint documents saved in the Office Open XML (OOXML) format for educational purposes. We briefly describe the structural ideas underlying the OOXML format and show examples of how certain document features are represented. Building on this, we describe the tool we developed for automatic evaluation of students' productions in the context of entrance examinations at our institution. The exam tests the participants' word-processing and spreadsheet abilities. We report the results of a case study comparing manual and automatic evaluation. We discuss the benefits (speed, cost, accuracy) and limitations (for situations where we lack precise formal conditions to test) of our approach.**

I. Introduction

The technology for complete, accurate, nuanced, automatic assessment of students' productions, if it existed, would be fantastic both in the classroom and for online offerings. Personalized and targeted feedback, from a learner's point of view, is an essential part of their path towards mastery of the topic of study [1]: when used appropriately, it can encourage engagement, catch common misconceptions early, and strengthen the intrinsic motivation towards the task. In practice, time and resource limitations prevent teachers from giving constant personal feedback in all but a few specific cases. With the advent of massive open online courses (MOOCs), the lack of personal feedback given by a human instructor is a reality. Most courses proposed on the Coursera and EdX platforms are free, which makes it unfeasible to hire more experts to assess and give feedback on the work on the thousands or tens of thousands of participants.

There are two categories of techniques that help address these issues (and which are extensively used in MOOCs): peer evaluation and automatic evaluation [2]. Peer evaluation comprises its lot of issues like seriousness of participants, lack of expertise of the evaluators, quality and equity control. Automatic evaluation delivers consistent and repeatable results practically instantly, treats all participants equally, can be repeated ad lib, and only costs human resources in the development phase, not during evaluation. However, it is often ill suited for the evaluation of work discussing open or subjective topics [3], and works best when the participants answer objective questions or produce work whose key points are technically quantifiable [4]. At any rate, it is inexorably making its way as an important component of learning technologies [5].

In this paper, we focus on the automatic extraction of formal properties of Microsoft Word, Excel, and PowerPoint documents for the assessment of basic office abilities. Such properties are e.g. formatting, alignment, use of tabs, headers, footers, footnotes, tables, etc. for Word; cell contents (values and formulas), chart format, axes properties for Excel; slide layout and formatting for PowerPoint.

This paper is structured as follows: in Section II, we discuss related work in metadata extraction. In Section III, we describe the relevant aspects of Office Open XML, the modern file format underlying the produced Office documents, and show how we can easily extract properties from it. We discuss a case study in Section IV—the technical entrance examination at our institution—which prompted us to look into this approach in the first place, and compare manual and automatic evaluation in terms of quality. We finally conclude in Section V.

II. Related Work

A lot of work has been produced in areas related to the extraction of information from word-processing documents. For instance, [6] deals with the automatic extraction of the title of a document, using both linguistic and formatting features. The automatic recognition and extraction of cited references is an important topic in the classification and referencing of scientific papers [7], [8].

Other approches include automatic creation of the summary of a word-processing document [9] and evaluation of those summaries [10]. Considerable effort has also been put into automatic essay scoring [3], [11], not without notable criticism against it [12], calling it unfair, inaccurate, reductive, or opaque [13].

We are interested in extracting well defined and quantifiable features for automatic processing. The closest to this we could find is the extension to the Moodle e-learning platform presented by [14], which is worth inspecting further. In the proposed approach, instructors can create auto-evaluated

assignments based on the presence or absence of a given structure in a document submitted by students. This structure is specified by a *regular expression* run against the XML text file that represent the main document. A regular expression is a sequence of characters which, essentially, specify a flexible text search pattern.[1] As the XML file contains not only all the text that the students type into their documents, but also the structure identifying formatting and layout (see the next section for details), this makes it possible to automatically check for both textual content and particular formatting or structure. While on the one hand, the Moodle extension offers a user-friendly front end to the creation of such assignments, writing regular expressions to match the correct structure can be hard—and re-reading and interpreting them is harder.[2]

To the best of our knowledge, that is the only published approach similar to the one presented here.

III. A Brief Anatomy of Office Open XML Documents

Since 2003, Microsoft Word, Excel, and PowerPoint documents can be saved in formats collectively known as Office Open XML (OOXML), following the Open Packaging Conventions [15]. Their file extensions are `.docx`, `.xlsx`, and `.pptx`, respectively. Essentially, an OOXML document is a ZIP archive[3] comprising various content:

- (a) a main XML file[4], serving as entry point for parsing the rest of the document;
- (b) relationship files, which are XML files listing and assigning a string ID to auxiliary files linked to from (a);
- (c) the actual auxiliary files referenced by (b). These auxiliary files can be embedded images or charts, or parts of the main document such as footnotes or styles, saved separately from the main file (a). They can also be whole new OOXML documents in their own right—which can in turn be recursively inspected and decomposed in the same way.

Any OOXML document can be manually renamed to `.zip` and decompressed, and the contained XML files can easily be inspected in a text editor.

For **Word** documents, the root file (a) in the archive is `word/document.xml`. The relationship file (b) resides in a sibling folder called `_rels` and is always named after the root file: here, `word/_rels/document.xml.rels`. Notable auxiliary files (c) include the list of footnotes, the various headers and footers that were defined, styles, and any embedded image or chart.

The root file of **Excel** documents is `xl/workbook.xml`, which links (through `xl/_rels/workbook.xml.rels`) to an auxiliary XML file for each worksheet, usually in the `xl/worksheet/` subdirectory. Sheet files contain an XML element for each nonempty cell, indicating its value and/or its formula as well as its formatting. There are auxiliary files for, notably, embedded images or charts.

Similarly, **PowerPoint** documents are read from the `ppt/presentation.xml` root file, down to the linked separate XML files for each slide. There are auxiliary files for shared slide layouts, master slides, general presentation properties, and for all embedded media files.

The XML schema definitions, listing all the XML element and attribute names used in the XML files, can be found in the standard, but can more easily be browsed through from a schema inspection website.[5]

For the sake of brevity, the next subsection focuses on the XML structure of the root Word file only, but similar observations can be made for the structure of Excel and PowerPoint XML files.

Structual Overview of `word/document.xml`

Here is a rough general structure illustrating how certain document features are saved in the root XML file of a `.docx` document.

- `<w:document>` is the root XML element, containing a `<w:body>` element, which itself contains a series of subelements. Most of these are paragraphs (`<w:p>`) or tables (`<w:tbl>`). A trailing element (`<w:sectPr>`) contains properties for the default document section.
- `<w:sectPr>` elements describe the paper size, margins, columns, and have potential references to headers and footers attached to the sections they define.
- `<w:p>` elements (paragraphs) mainly contain `<w:r>` elements (text runs). When a saved document includes revision information, paragraphs also contain `<w:ins>` and `<w:del>` elements, representing inserted and deleted text runs, respectively.
- `<w:pPr>` elements contain paragraph properties: paragraph style name and/or list style name (referenced from a linked file describing the properties of said style), before/after spacing, first-line indentation, text alignment and justification, borders, tab stops and tab types, etc.
- `<w:r>` elements contain `<w:t>` elements (`t` for "text"). These have as XML text content the actual text that was typed into the Word document. `<w:t>` elements follow each other inside a `<w:r>`, and `<w:r>` elements follow each other inside a `<w:p>`. New `<w:r>` elements are inserted in a `<w:p>` at least when the style changes, but also seem to be sometimes inserted by Word for contiguous, same-style text. `<w:r>` elements also contain other elements signalling footnote or endnote references, inserted

[1] See http://regexone.com for more info and for an interactive tutorial on regular expressions.

[2] A 1997 quote by Jamie Zawinski, an early Netscape engineer, got famous: "Some people, when confronted with a problem, think 'I know, I'll use regular expressions.' Now they have two problems."

[3] ZIP is a popular file compression format. It can be used to create a single, compressed file from a given folder containing files and subfolders. Thus, OOXML files are actually compressed archives of several files.

[4] XML is a text-based markup language that uses hierarchical custom tags, like `<text>` or `<footnote>`, to structurally organize data. A simple example file can be viewed at http://www.w3schools.com/xml/note.xml.

[5] For instance, http://www.schemacentral.com/sc/ooxml/ss.html lists the whole of OOXML, and http://www.schemacentral.com/sc/ooxml/s-wml.xsd.html lists the "vocabulary" used in `.docx` files.

symbols, tabs, comments, line/column/page breaks, and inserted objects like drawings or charts.

- <w:rPr> elements contain text run properties, like character style name (also referenced from a linked file), font name, weight, variant, foreground and background colors, text effects, borders, etc. (Notice the pattern where <*x*Pr> is the element containing properties of the enclosing <*x*> element.)
- Similarly, <w:tbl> elements contain table properties (<w:tblPr>) and a certain number of rows (<w:tr>), each of which contains cells (<w:tc>). Cells in turn contain cell properties (<w:tcPr>, indicating borders, background color, possibly merged cells status, etc.) and finally one or more <w:p> paragraphs.

The general structure of the document is clearly and very readably expressed with these XML elements. They offer an easily accessible description of most aspects of the document (although more delicate cases will be mentioned below).

Having realized this, we decided to build an automatic correction tool for the technical entrance examination at our institution, described in the next section.

IV. Case Study: Technical Entrance Examination at HEP-VD

The Lausanne University of Teacher Education (HEP-VD) is attended by future teachers (hereafter, referred to as *students*) of primary school (age 4–12), secondary school (age 12–16), and high school (age 16–19). A few years ago, it was decided to introduce an entrance examination to ensure a minimum level of mastery of (a) French, the native language of most pupils the students would teach; and (b) basic computer software that would be used extensively during courses at HEP-VD—and most probably during the students' future professional practice. Each year in September, about 700 new students must take this combined exam. Two further sessions are organized in January and June, where the attendance averages respectively 100 and 30 students passing the exam again.

We were involved with the realization of (b), which was designed in two subparts: a multiple-choice questionnaire (whose automatic correction is straightforward), and a practical part consisting of the reproduction, with word-processor and spreadsheet software, of a two-page printed template document that includes a table, a graph, and a reasonable variety of styles and alignments. This document is typically meant to look like an elaborate document a teacher would hand out to his pupils.

While the reproduction of the text itself poses little or no challenge, the adequate use of software features are the main points that are tested in this second part. Here are the precise criteria that award points (this list is also available to students before and during the exam):

1) Setting the document's margins;
2) Inserting a header and a footer, one of them with a page number and a border;
3) Defining tab stops and consistently using tabs instead of multiple spaces;
4) Inserting symbols or special characters which cannot directly be typed with a single keystroke—for instance, É or ë in French, or other Unicode glyphs like ✂ or ✎;
5) Using leader characters for tabs to create visual guides or fields to fill in in a form (instead of e.g. inserting repeated dots or underscores);
6) Inserting a page or column break;
7) Centering and justifying paragraphs as in the printed template;
8) Inserting a footnote, not simulated by a text box;
9) Inserting a table, setting cells' background colors, merging cells horizontally or vertically;
10) Aligning the contents of cells vertically and horizontally;
11) Using paragraph before/after spacing instead of multiple carriage returns;
12) Using first-line indentation: negative for lists or definitions, or positive for text paragraphs;
13) Inserting an image from a given file or URL and adding a text box below it for the image caption;
14) Making the text wrap around the image and text box;
15) Creating a graph and inserting it into the main document;
16) Setting the graph's title, legend, axis names, axis labels;
17) Highlighting some part of the graph with a custom drawing (oval or arrow);
18) Laying out the whole document on two pages according to the template, globally respecting the relative positions of the various elements (paragraphs, tables, images, and charts).

For several years, this exam was evaluated by hand: correctors would open the produced files and manually verify each criterion described above to decide whether or not to award the point. The submitted student documents were divided into series of about 60 documents for each corrector, who had to correct all criteria for those documents. Not counting occasional mistakes, it turned out that some of these criteria could be interpreted slightly differently between correctors, and thus, points could be awarded inconsistently. Some correctors were also occasionally fooled by a student's inadequate but very similar-looking fallback solution: for instance, manually inserted "page numbers" (that would not have worked for additional pages any more), headers inserted as text boxes instead of being genuine headers, or vertical cell alignment simulated by carriage returns. In addition to that, finding the dozen needed qualified correctors could be a challenge in itself, and the whole correction process was costly and lengthy, lasting several weeks until all results were checked in and finally communicated to the students.

In early 2014, we decided to develop with the Scala programming language[6] a tool to automatically extract and inspect the XML structure of the produced documents. Scala is a concise, statically typed language that runs on the Java Virtual Machine (JVM) and which has language-level support for XML. Virtually all operating systems used in our institution have a JVM preinstalled. Choosing a technology that runs on the JVM also made it possible to easily reuse our tool (or parts of it) in another context—for instance, on a webpage accessible by students before the exam to give them an opportunity to train on mock exams testing similar criteria.

[6] http://www.scala-lang.org

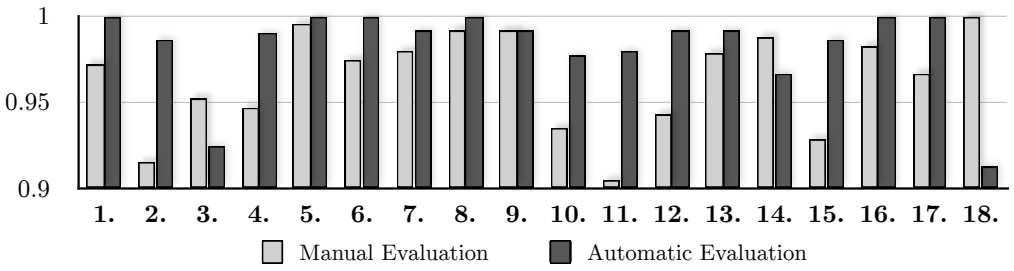

Fig. 1. Plot of the unrounded F_1 score for each criterion, for manual and automatic evaluation. Higher scores denote better accuracy. A score of 1 signifies the absence of both false positives and false negatives and thus a perfect evaluation method with respect to the gold standard.

Most of the listed criteria can be evaluated quite precisely in a dozen lines of Scala code. That tool could be given as input a series of documents and would output an Excel summary of the points obtained by each student. (As students were required to type in their name in the header of the produced document, the tool could extract the students' names directly from the produced document as well.)

In the next subsection, we quantitatively discuss the differences in quality and sensitivity between manual evaluation and automatic evaluation with our tool.

Automatic vs. Manual Evaluation

We compared the difference in quality between automatic and manual evaluation on a sample of 127 student documents. These documents had initially been corrected manually according to our standard procedure, were then run through the automatic corrector, and were again recorrected manually twice with utmost care consistently by a single person to establish the reference "gold standard." To compare the results, we used measures known as *precision and recall* in pattern recognition and information retrieval, and a compound measure known as the F *score*. They each are measures of the goodness of the corrector for a given criterion and of the number of mistakes that it made with respect to the gold standard (see, e.g., [16], pp. 182–186).

In our context, such mistakes can be of two kinds. A *false positive* occurs when the correction method awards a point for a criterion that was not fulfilled. Conversely, a *false negative* occurs then the correction method fails to award a point when it was deserved. Following that, the precision p is a number between 0 and 1 indicating, for a given criterion c, the fraction of documents where c was correctly evaluated as being fulfilled with respect to the gold standard. When $p = 1$, there are no false positives and the evaluation method never awards more points than it should. Similarly, the recall r for a criterion c indicates, out of all documents where c holds according to the gold standard, the fraction that was effectively evaluated as such. When $r = 1$, no there are no false negatives and no points are left out by the evaluation method.

Precision and recall should be considered together, as it is easy to maximize one without considering the other: to reach perfect precision, we could simply award zero points at all; to reach perfect recall, we could award them all.

The F score is itself a combination of precision and recall. In its general form, it is parameterized by a non-negative value for β in this definition [17]:

$$F_\beta = \frac{(1+\beta^2)\cdot \text{precision}\cdot \text{recall}}{\beta^2\cdot \text{precision} + \text{recall}}. \tag{1}$$

Like precision and recall, F_β is always between 0 and 1 (both inclusive). The β parameter allows us to choose how much more importance we want to give to recall with respect to precision. Examples are the F_2 score ($\beta = 2$), which gives a combined measure where recall is twice as important as precision; the $F_{0.5}$ score, where precision is twice as important as recall; and the balanced F_1 score, which gives as much importance to precision as to recall.

In the context of the evaluation of our correction methods, we had no particular reason to favour either precision or recall, and chose the F_1 score, which, according to 1, then simplifies to:

$$F_1 = \frac{2\cdot \text{precision}\cdot \text{recall}}{\text{precision} + \text{recall}}. \tag{2}$$

A perfect evaluation method making no errors whatsoever would thus have scores $p = r = F_1 = 1$.

TABLE I.
ROUNDED PRECISION p, RECALL r, AND F_1 SCORE FOR EACH CRITERION, FOR MANUAL AND AUTOMATIC EVALUATION.
THE BETTER F_1 SCORES HAVE A LIGHT GRAY BACKGROUND.

Criterion		**1.**	**2.**	**3.**	**4.**	**5.**	**6.**	**7.**	**8.**	**9.**	**10.**	**11.**	**12.**	**13.**	**14.**	**15.**	**16.**	**17.**	**18.**
Manual Evaluation	p	0.97	0.84	0.95	0.90	0.99	0.95	0.97	1	0.98	0.93	0.94	0.96	0.96	0.99	0.87	0.97	0.96	1
	r	0.97	1	0.95	1	1	1	0.99	0.98	1	0.94	0.87	0.92	1	0.98	1	1	0.98	1
	F_1	0.97	0.91	0.95	0.95	0.99	0.97	0.98	0.99	0.99	0.93	0.91	0.94	0.98	0.99	0.93	0.98	0.97	1
Automatic Evaluation	p	1	0.98	0.92	0.99	1	1	1	1	1	0.97	0.96	0.98	0.98	0.95	0.99	1	1	0.93
	r	1	0.99	0.93	0.99	1	1	0.98	1	0.98	0.98	1	1	1	0.98	0.98	1	1	0.90
	F_1	1	0.99	0.92	0.99	1	1	0.99	1	0.99	0.98	0.99	0.99	0.99	0.97	0.99	1	1	0.91

For each numbered criterion listed above, we show the precision, recall, and F_1 score numerically in Table I, and the F_1 score graphically in Figure 1 for the manual and automatic correction.

These results show that the automatic correction yields equal or more accurate results than the manual evaluation on all but 3 of the 18 criteria, as measured by the F_1 score. On the other hand, these numbers beg the question: why wouldn't an automatic evaluation always have perfect scores in the first place?

Of course, the software tool will never fail to identify a footnote or the exact spacing around a paragraph. But our gold standard here is actually a careful manual evaluation which does not exclusively focus on the technical aspects of a criterion, but also takes into account its context and tries to respect the global meaningfulness of the assignment. Take Criterion 3, for instance, where we test the use of tabs. Our aim is to ensure that the students know when to use tabs and will not insert multiple spaces to simulate them. Now, in this context, would it be a mistake to insert two tabs in a row, instead of a single tab with a redefined tab stop? The answer depends on the context—it may or may not make sense. Most students are neither IT professionals nor book editors; the evaluation of this exam should not needlessly penalize them because of the overspecificity of too stringently defined criteria. Defining the exact formal conditions where it does or does not make sense can be challenging.

Other examples that explain why the automatic evaluation does not yield perfect scores include slightly inconsistent but barely noticeable paragraph spacing, cell alignment, text wrapping: in each case, a manual evaluation has the opportunity to be more sensitive about non-erroneous, context-sensitive departures from the exact expected structure, where the automatic evaluation will issue a verdict blind to the context.

Criterion 18 deserves a special note, as the natural (non-forced) page break information is not always saved in the document. A `<w:lastRenderedPageBreak>` element does exist, but is to be treated as a hint: its absence does not imply that the surrounding text fits on a single page. As surprising as it may seem, we therefore do not directly know how many pages a document contains.

A further observation regarding the implementation of the automatic evaluation is the chosen tradeoff between the generality of a given criterion and its specific realization in the template exam document. For instance, a test to check how many paragraphs define 3 tab stops at given positions with a certain tolerance is easy to implement and can be general enough to be reused for future exams, but makes it theoretically possible to construct a meaningless document that would obtain the point. On the other hand, a test that checks if exactly x tabs were used in paragraphs y and z between the n^{th} and the m^{th} words makes it more "foolproof," but becomes overly sensitive to little variations in the produced document. This approach would moreover require a reimplementation (including careful testing) for each new template document.

That tradeoff is similar to the phenomenon known as *overfitting* in machine learning (see, e.g., [18], p. 6). Overfitting occurs when a statistical model or classifier becomes too complex with respect to the data it should operate on and has poor generalization capabilities with new data it has never seen. Roughly spoken, this happens when a model is let to focus too much on the peculiarities of its training data (and the potential noise in it) rather than on extracting the overall trends and relationships the data carries. Transposed to our situation, the automatic corrector could be modified to achieve better scores on the criteria where it can still make mistakes, but this modified corrector would be likely to overfit the specific template document used in this comparison and would be likely to perform less well on future, similar but not identical template documents—unless modified and tested again with those new documents.

Should we then be more precise in the definition of the criteria, for the sake of a more accurate automatic evaluation? We can be, as long as it does not impair the robustness of the correction tool across documents, and, more importantly, that the exam as a whole retains its global meaningfulness. While it is meaningful to require students to know how to define and use tab stops, it is not meaningful to demand that the tabs stops be defined for paragraphs y and z exactly and for no others.

In practice, we have chosen not to fundamentally alter our criteria, and have introduced an extra manual verification step for documents of students whose final overall scores are close to the pass/fail limit.

V. Conclusion & Outlook

Switching from a manual to an automatic evaluation of our technical, word-processing-based entrance examination at HEP-VD allowed us to shorten the assessment process from several weeks and more than a dozen correctors to just over one day with a single supervisor. The change also increased the quality of the evaluation and allowed for more resources being devoted to quality control—all without forcing us to overhaul our evaluation criteria.

Going automatic opens the door to new options. Previously ungraded aspects (e.g., details in the generation of the graph from the spreadsheet application, use of formulas in the spreadsheet, or other formal aspects that would take too much time for a human to grade) can now become part of the evaluation. Certain criteria could also be refined so as to allow for a continuous score from 0 to 1 instead of a binary score, without endangering the equality of treatment among students. One other significant possibility that automatic evaluation brings is a continuous, real-time, formative assessment of the students [19] with formative feedback during the exam itself as the documents are being constructed.

Finally, we could consider coupling the extraction of features from Office productions with an automated tutoring system that would support learners in their actions by incentives. The teachers expertise would then be directed to more complex tasks for which computer automation yet falls short.

Acknowledgments

We would like to thank Bernard Baumberger, Josiane Chevalley-Roy, and Emmanuel Flaction, head and members of our teaching and research unit, respectively, for establishing the initial form of the entrance examination described above and for providing past data of manually corrected copies to compare against.

REFERENCES

[1] H. Astleitner and J. M. Keller, "A model for motivationally adaptive computer-assisted instruction," *Journal of Research on Computing in Education*, vol. 27, no. 3, pp. 270–280, 1995.

[2] N. B. Shah, J. K. Bradley, A. Parekh, M. Wainwright, and K. Ramchandran, "A case for ordinal peer-evaluation in MOOCs," in *NIPS Workshop on Data Driven Education*, 2013.

[3] J. Wang and M. S. Brown, "Automated essay scoring versus human scoring: A comparative study," *Journal of Technology, Learning, and Assessment*, vol. 6, no. 2, pp. 1–28, 2007.

[4] K. Scalise and B. Gifford, "Computer-based assessment in e-learning: A framework for constructing "intermediate constraint" questions and tasks for technology platforms," *Journal of Technology, Learning, and Assessment*, vol. 4, no. 6, pp. 1–44, 2006.

[5] R. E. Bennett, "Inexorable and inevitable: The continuing story of technology and assessment," *Journal of Technology, Learning, and Assessment*, vol. 1, no. 1, pp. 1–23, 2002.

[6] Y. Hu, H. Li, Y. Cao, L. Teng, D. Meyerzon, and Q. Zheng, "Automatic extraction of titles from general documents using machine learning," Microsoft Research, Tech. Rep. MSR-TR-2006-17, 2006.

[7] D. Bergmark, "Automatic extraction of reference linking information from online documents," Cornell Digital Library Research Group, Tech. Rep. CSTR 2000-1821, 2000.

[8] S. Hitchcock, T. Brody, C. Gutteridge, L. Carr, W. Hall, S. Harnad, D. Bergmark, and C. Lagoze, "Open citation linking: The way forward," *D-Lib Magazine*, vol. 8, no. 10, 2002.

[9] I. Mani and M. T. Maybury, Eds., *Advances in Automatic Text Summarization*. MIT Press, 1999.

[10] C.-Y. Lin, G. Cao, J. Gao, and J.-Y. Nie, "An information-theoretic approach to automatic evaluation of summaries," in *HLT-NAACL*, 2006.

[11] S. Dikli, "An overview of automated scoring of essays," *Journal of Technology, Learning, and Assessment*, vol. 5, no. 1, pp. 1–35, 2006.

[12] P. Deane, "On the relationship between automated essay scoring and modern views of the writing construct," *Assessing Writing*, vol. 18, no. 1, pp. 7–24, 2013.

[13] R. Haswell and M. Wilson. (2013) Professionals against machine scoring of student essays in high-stakes assessment. http://humanreaders.org.

[14] R. Gérin-Lajoie and G.-P. Leblanc, "Automatic evaluation of Excel and Word productions in Moodle," in *Canada Moodle Moot*, 2013, available at http://www.slideshare.net/gpleblanc/2013-02-15prsentationcertitudemoodlemoot.

[15] "ECMA-376: Office Open XML file formats," http://www.ecma-international.org/publications/standards/Ecma-376.htm, 2006.

[16] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[17] C. J. van Rijsbergen, *Information Retrieval*. Butterworth-Heinemann, 1979.

[18] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2007.

[19] W. J. Popham, *Transformative assessment*. Association for Supervision and Curriculum Development, 2008.