bioPDFX: preparing PDF scientific articles for biomedical text mining

Shitij Bhargava [1] , **Tsung-Ting Kuo** [2] , **Ankit Goyal** [1] , **Vincent Kuri** [1] , **Gordon Lin** [1] , **Chun-Nan Hsu** Corresp. [2]

[1] Department of Computer Science and Engineering, Jacobs School of Engineering, University of California, San Diego, La Jolla, California, United States

[2] Health System Department of Biomedical Informatics, School of Medicine, University of California, San Diego, La Jolla, California, United States

Corresponding Author: Chun-Nan Hsu
Email address: chunnan@ucsd.edu

Background. There is huge amount of full-text biomedical literatures available in public repositories like PubMed Central (PMC). However, a substantial number of the papers are in Portable Document Format (PDF) and do not provide plain text format ready for text mining and natural language processing (NLP). Although there exist many PDF-to-text converters, they still suffer from several challenges while processing biomedical PDFs, such as the correct transcription of titles/abstracts, segmenting references/acknowledgements, special characters, jumbling errors (the wrong order of the text), and word boundaries.

Methods. In this paper, we present bioPDFX, a novel tool which complements weaknesses with strengths of multiple state-of-the-art methods and then applies machine learning methods to address all issues above

Results. The experiment results on publications of Genome Wide Association Studies (GWAS) demonstrated that bioPDFX significantly improved the quality of XML comparing to state-of-the-art PDF-to-XML converter, leading to a biomedical database more suitable for text mining.

Discussion. Overall, the whole pipeline developed in this paper makes the published literature in form of PDF files much better suited for text mining tasks, while slightly improving the overall text quality as well. The service is open to access freely at URL: http://textmining.ucsd.edu:9000 . A list of PubMed Central IDs of the 941 articles (see Supplemental File 1) used in this study is available for download at the same URL. The instructions of how to run the service with a PubMed ID are described in Supplemental File 2.

bioPDFX: preparing PDF scientific articles for biomedical text mining

Shitij Bhargava[1], Tsung-Ting Kuo[2], Ankit Goyal[1], Vincent Kuri[1], Gordon Lin[1], Chun-Nan Hsu[2]

[1] Department of Computer Science and Engineering, Jacobs School of Engineering, University of California, San Diego, La Jolla, California, United States

[2] Health System Department of Biomedical Informatics, School of Medicine, University of California, San Diego, La Jolla, California, United States

Corresponding Author:

Chun-Nan Hsu

9500 Gilman Drive, La Jolla, CA 92093, United States

Email address: chunnan@ucsd.edu

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

Abstract

Background. There is huge amount of full-text biomedical literatures available in public repositories like PubMed Central (PMC). However, a substantial number of the papers are in Portable Document Format (PDF) and do not provide plain text format ready for text mining and natural language processing (NLP). Although there exist many PDF-to-text converters, they still suffer from several challenges while processing biomedical PDFs, such as the correct transcription of titles/abstracts, segmenting references/acknowledgements, special characters, jumbling errors (the wrong order of the text), and word boundaries.

Methods. In this paper, we present bioPDFX, a novel tool which complements weaknesses with strengths of multiple state-of-the-art methods and then applies machine learning methods to address all issues above.

Results. The experiment results on publications of Genome Wide Association Studies (GWAS) demonstrated that bioPDFX significantly improved the quality of XML comparing to state-of-the-art PDF-to-XML converter, leading to a biomedical database more suitable for text mining.

Discussion. Overall, the whole pipeline developed in this paper makes the published literature in form of PDF files much better suited for text mining tasks, while slightly improving the overall text quality as well. The service is open to access freely at URL: http://textmining.ucsd.edu:9000. A list of PubMed Central IDs of the 941 articles (see Supplemental File 1) used in this study is available for download at the same URL. The instructions of how to run the service with a PubMed ID are described in Supplemental File 2.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

1. Introduction

PubMed Central (PMC) (2015e) contains full-text of about 4 million biomedical literatures, and is one of the most important freely available data sources for the biomedical natural language processing (NLP) research field. Although many new publications now provide plain text along with the Portable Document Format (PDF), a substantial number of them do not. However, text mining algorithms work more effectively on text-based formats such as plain text, XML or HTML documents. Therefore, NLP researchers usually need to transcribe the biomedical literatures from PDF to text as a preprocessing step of the NLP pipeline.

However, transcribing PDF to text accurately is not trivial. The design goals of the PDF standard target ease of human readability rather than electronic consumption of data by other software tools. That is, the PDF standard does not attempt to encode any semantic connection between characters in a word or between paragraphs, but characters are “painted" individually at specific 2-dimensional coordinates. Berg et. al. outlined some challenges encountered in transcribing PDF documents to text accurately, such as varied reading orders, sectional formatting and number of columns (Berg 2011). These issues create a variability in text quality in the generated outputs and impedes accurate text mining.

Although there are already a variety of tools available today for PDF-to-text transcription (such as Apache PDFBox Mozilla PDF.js (2015d) Adobe Acrobat SDK (2015a) Tesseract Optical Character Recognition (OCR) (Smith 2007) (Constantin et al. 2013)several challenges to be solved for converting biomedical literature PDF to XML for text mining purpose::

1. *Extracting title and abstracts correctly*. In case of manuscripts which might have contents other than title or abstract on their first page, conversion tools like PDFX struggle to

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

identify them correctly.

2. *Identifying reference and acknowledgement sections in various format*. The format of references varies considerably among different publications ranging from a separate section towards the end to individual references in footnotes continuing alongside the content body. Take PDFX as an example, as a purely rule-based system, it often falters in recognizing references when the common pattern of having references in the end with a heading is not present. Similar problems were noticed in identifying acknowledgements as well.

3. *Recognizing special characters (such as '!' and '@')*. PDFX and all PDF to text converters usually suffer from the problem of recognizing the special characters depending on the publishing of the PDF files.

4. *Detecting and fixing jumbling errors (the wrong order of text while converting PDF to text)*. Often most PDF conversion tools jumbles the work order in or across sentences due to mistakes in reading order detection/segmentation. An example is given in Figure 1 where the exponents of numbers in a table are extracted as a separate column and hence are jumbled with respect to the actual text.

5. *Correcting word boundaries*. Often words straddling the column boundary have a hyphen inserted in between them (e.g., “cancer” might become “can-” and “cer” if it is the last word of a row). Although this is not an error due to conversion, we would still like to correct it and merge the hyphenated parts as a single word wherever possible to support NLP afterwards.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

Figure 1. A kind of jumbling error in a table.

The table at the top is what is present in the PDF. The lower portion shows the transcribed XML. The exponents marked in the red-colored box are incorrectly extracted as a separate column in the XML.

A. Discovery group.

dbSNP	Chromosome	Physical Position	Gene Relationship	Gene Distance	Gene	BP Change	Prob Allele	Prob Genotype
rs6975107	7	120168143	intron	0	KCND2	C-> T	$4.15x10^{-09}$	$3.74x10^{-09}$
rs318125	X	97090551	downstream	348,298	DIAPH2	C-> T	$4.35x10^{-09}$	$1.25x10^{-05}$
rs5916727	X	104294270	intron	0	IL1RAPL2	C-> T	$6.66x10^{-09}$	$1.34x10^{-05}$
rs11863929	16	86861934	upstream	159,445	ZNF469	C-> G	$1.77x10^{-08}$	$5.48x10^{-07}$
rs1578826	X	85889112	intron	0	DACH2	T-> C	$2.20x10^{-08}$	$5.46x10^{-06}$
rs7616661	3	5940543	downstream	703,894	EDEM1	T-> G	$4.82x10^{-08}$	$3.37x10^{-08}$

```
<region class="DoCO:TextChunk" id="586" confidence="possible" page="26"
column="1">A. Discovery group.</region>
<outsider class="DoCO:TextBox" type="sidenote" id="587" page="26"
column="1">-09 -05 -05 -07 -06 -08</outsider>
-<region class="unknown" id="588" page="26" column="1">
   -09 rs6975107 7 120168143 intron 0 KCND2 C-> T 4.15x10 3.74x10 -09 rs318125 X
   97090551 downstream 348,298 DIAPH2 C-> T 4.35x10 1.25x10 -09 rs5916727 X
   104294270 intron 0 IL1RAPL2 C-> T 6.66x10 1.34x10 -08 rs11863929 16 86861934
   upstream 159,445 ZNF469 C-> G 1.77x10 5.48x10 -08 rs1578826 X 85889112 intron 0
   DACH2 T-> C 2.20x10 5.46x10 -08 rs7616661 3 5940543 downstream 703,894 EDEM1
   T-> G 4.82x10 3.37x10
</region>
```


PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

To address these issues, we proposed bioPDFX, a novel tool that integrates current state-of-the-art PDF conversion methods to transcribe biomedical PDF articles to high quality text in XML format. The bioPDFX tool leverages the following four tools to transcribe text of the given PDF: *PDFX* (Constantin et al. 2013), *PMC Entrez e-utilities API* (Sayers et al. 2011), *Tesseract OCR* (Smith 2007), and *Apache PDFBox* (2015b). The input of bioPDFX includes a biomedical literature PDF file and its corresponding PubMed ID, while the output is the converted XML file. The overall processing pipeline of bioPDFX consists of the following five components, each address an abovementioned challenge:

1. *Title and Abstract Correction*. We used the PMC Entrez e-utilities API to retrieve the correct title and abstract from PubMed.
2. *Reference and Acknowledgment Correction*. To detect whether a paragraph is reference, acknowledgement, or none of them, we designed two binary classifiers, one for reference and the other for acknowledgement. For reference detection, we extracted features such as density of year (e.g., "2016"), density of "et. al.", and density of numbers followed by the dot (e.g., "12.". We normalized these features and train a classifier to predict whether a paragraph is a reference text. For acknowledgment detection, we used a classifier with features being TF-IDF followed by Latent Semantic Analysis (Deerwester et al. 1990) to perform the detection.
3. *Special Character Correction*. The basic idea is to compare the results from PDFX and that from Tesseract OCR to identify suspicious characters (i.e., different identified characters within the same n-gram), and then apply Hidden Markov Model (HMM) (Baum & Petrie 1966) with Viterbi inference algorithm (Viterbi 1967) and language model to choose the "most probable" candidate character for the mismatch character by maximizing

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

the overall likelihood to recover the n-gram.

4. *Jumbling Error Correction*. We used the text extracted using Apache PDFBox to correct the possible jumbling errors from PDFX. We first detected jumbled n-gram, and then replaced them by the non-jumbled version from Apache PDFBox.

5. *Word Boundary Correction*. We adopted a simple dictionary lookup method by comparing different parts (e.g., "can-" and "cer") with the English dictionary in the Enchant Spellchecking System (2015c) and merging them as required.

To evaluate the bioPDFX tool, we randomly extracted 100 biomedical literatures related to Genome Wide Association Study (GWAS) (Hindorff et al. 2009; Welter et al. 2014) from PMC, and used XML versions (i.e., NXMLs) of those 100 articles provided in PMC as gold standards. We compare bioPDFX with the state-of-the-art PDFX tool (Constantin et al. 2013). As a use case, we also compare the results of converting important information (p-value and number) and overall text quality in the GWAS papers using bioPDFX and PDFX. We evaluate both individual correction steps and overall conversion results. The experiment results show significant improvement for all five types of correction tasks, especially for abstract, reference/acknowledgement, special character, and jumbling error corrections.

Our contributions of this study are three-fold:

- We identified five issues (title/abstract, reference/acknowledgment, special character, jumbling error, and word boundary) of the state-of-the-art PDF-to-XML tools as the very first step of biomedical NLP pipeline.

- We integrated the output of four popular tools (PDFX, PMC Entrez e-utilities API, Tesseract OCR, and Apache PDFBox) in bioPDFX and designed five corresponding

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

correction steps to solve the abovementioned issues.

- We evaluated bioPDFX and compare it to PDFX for each correction steps as well as a real-world GWAS information extraction task, and the results demonstrated that bioPDFX can improve all the five issues and increase the extraction accuracy for GWAS literatures.

Related Work. The rest of the section reviews the related studies along with the mention of tools that we use in our approach. PDF text extraction involves extracting out all textual content from the PDF file in an order that makes sense even without the formatting of the PDF. A variety of tools are available that parse and convert PDF to text such as Apache PDFBox (2015b), Mozilla PDF.js (2015d), and Adobe Acrobat SDK (2015a). Most text extractors use a page segmentation algorithm to determine all the textual areas of the PDF based on width of the whitespace and from these boundaries also determine a reading order. A major problem associated with the text based extraction is because of variability of PDF publishing software. Some tools embed a special character that is not included in a standard font by drawing a vector graphics over it. Thus, although it might look perfectly correct to a reader, the information is not present in the text layer of the PDF for text extractors to extract. In our bioPDFX tool, we integrated text extraction from Apache PDFBox with other tools such as PDFX and optical character recognition tool Tesseract OCR to overcome challenges like this.

XML transcription of a PDF file involves extraction of text, tables, images, etc. and then tagging them with appropriate sections/regions to make an XML. The exact format of the transcribed XML depends on the specific tool and might have a tag for title, abstract, introduction, tables, etc. Some tools like PDFX (Constantin et al. 2013) use a schema similar to JATS/NLM document-type definition (DTD) format which is a specific format for scientific articles and are

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

158 highly optimized around the organization of such PDF files. PDFX is a freely available rule-based
159 system which converts scholarly PDF articles to XMLs which have detailed sectional information
160 such as title, abstract, references, tables, and acknowledgment. It derives transcription parameters
161 from the relative font sizes, style and spacing of articles with respect to individual PDF files. In
162 our work, we integrate PDFX with other extraction tools to improve the extraction results for texts
163 like title, abstract, reference, and acknowledgement.

164 Text post-processing is a step that improves the correctness of the generated text. Several
165 tools, such as Tesseract OCR (Smith 2007), deploy techniques from Optical Character Recognition
166 (OCR) to increase the accuracy in the post-processing stage. Because most OCR engines classify
167 each character independent of other characters, it is important to post-process the text using some
168 contextual information such as statistical language models or syntactic and semantic rules. One
169 such method suggested by Zhuang and Zhu (Zhuang & Zhu 2005) reduces the candidate character
170 search space for characters by using the candidate distance information by an OCR engine in
171 conjunction with an n-gram based language model and a semantic lexicon. In another method
172 proposed by Velagapudi (Velagapudi 1999), contextual information was extracted by modeling
173 words/n-grams as sequences of characters and a Hidden Markov Model (HMM) was subsequently
174 used to decide the best sequence, using the given OCR output as the observation. Transition
175 probabilities of one character to another are calculated from a corpus and emission probabilities
176 are calculated from the output of the OCR engine on a corpus, to statistically record the error
177 patterns of the OCR engine. At the word level, Tong and Evans (Tong & Evans 1996) corrected
178 OCR text by modeling the text as sequence bigrams and then using HMM to compute the best
179 word sequence. The system learns the character level confusion probabilities for a specific OCR
180 engine and uses it to achieve better performance. In our study, we follow a similar approach to

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

post-process the output from PDFX using the Tesseract OCR engine.

In addition to above methodologies, several systems extract metadata such as authors, year of publication, journal name, and bibliographical information from natural language text or PDFs directly. CERMINE (Tkaczyk et al. 2014) uses supervised and unsupervised machine learning techniques to extract parsed bibliographic references and metadata directly from PDFs. FLUX-CiM (Cortez et al. 2007) uses unsupervised, non-template methods to extract citation components from articles. Other PDF transcribers include LA-PDFText.(Ramakrishnan et al. 2012) Although the abovementioned tools are available for PDF-to-text transcription, several issues are yet to be dealt with, such as how to extract title and abstracts correctly, identify reference acknowledgement sections in various format, recognize special characters correctly, detect and fix jumbling errors, and correct word boundaries. PDFJailbreak (Garcia et al. 2013) is a communal project to create an architecture and shared API to support open community development of semantic information extractor from biomedical literature in PDF form.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

2. Materials and Methods

The goal of this study is to extract high-quality XMLs from a biomedical literature PDFs to make it better suited for our text mining tasks. Given a PDF file and its corresponding PubMed ID, our system integrates the following four tools to generate output XML files:

- *PDFX* (Constantin et al. 2013) to transcribe XML of the given PDF
- *PMC Entrez e-utilities API* (Sayers et al. 2011) to retrieve text from PubMed
- *Tesseract OCR* (Smith 2007) to extract text extracted from the PDF file
- *Apache PDFBox* (2015b) to transcribe text of the given PDF

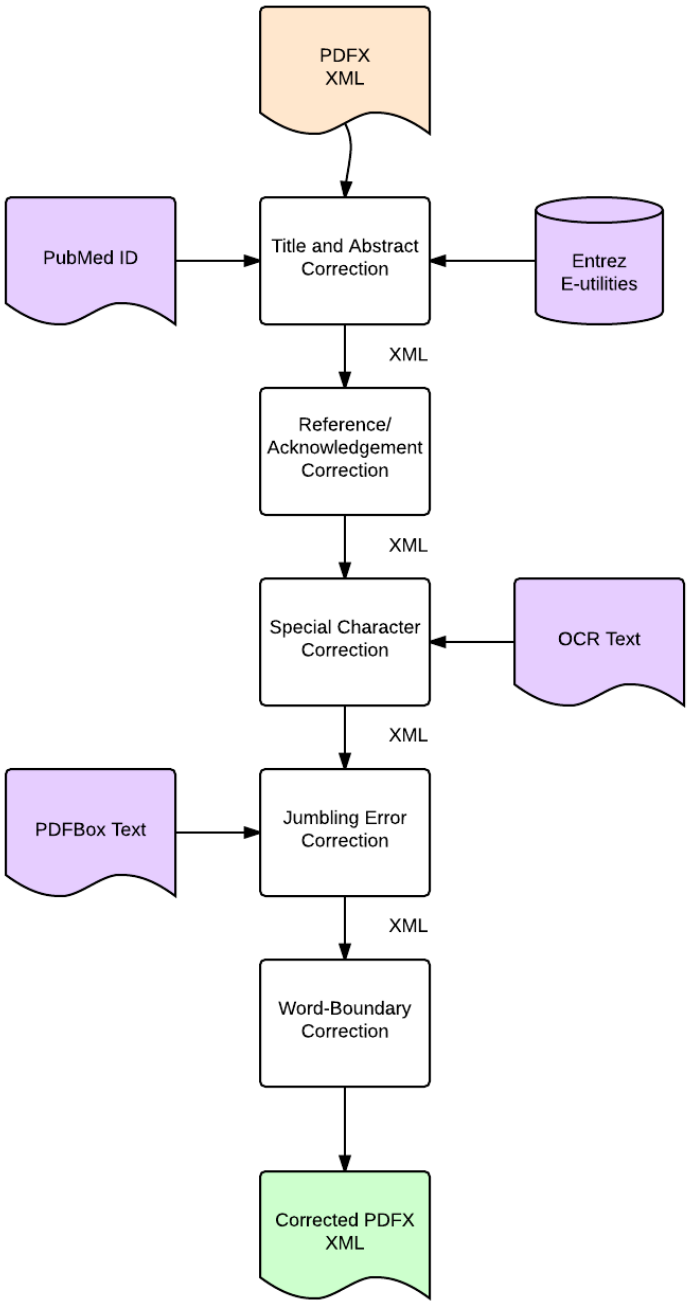
As shown in Figure 2, there are five correction stages (title/abstract, reference/acknowledgment, special character, jumbling error, and word boundary corrections) in the bioPDFX pipeline. Each stage is a filter and accepts an XML as input, giving a corrected XML as output. We start from the XML generated by PDFX, correct title and abstract using PMC Entrez e-utilities API, identify reference and acknowledgement texts, exploit Tesseract OCR to fix special characters, utilize Apache PDFBox to correct jumbling errors, and recover word-boundaries as our final stage. It should be noted that the pipeline design of bioPDFX allows different order of correction stages, as well as adding new correction stages. The detail of each of the correction stages is discussed in the following subsections.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

215 **Figure 2. A flow diagram showing the stages of correction and their inputs.**

216 Stages are arranged in a pipelined fashion and take an XML as input and return a corrected XML

217 as output.



218

219

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

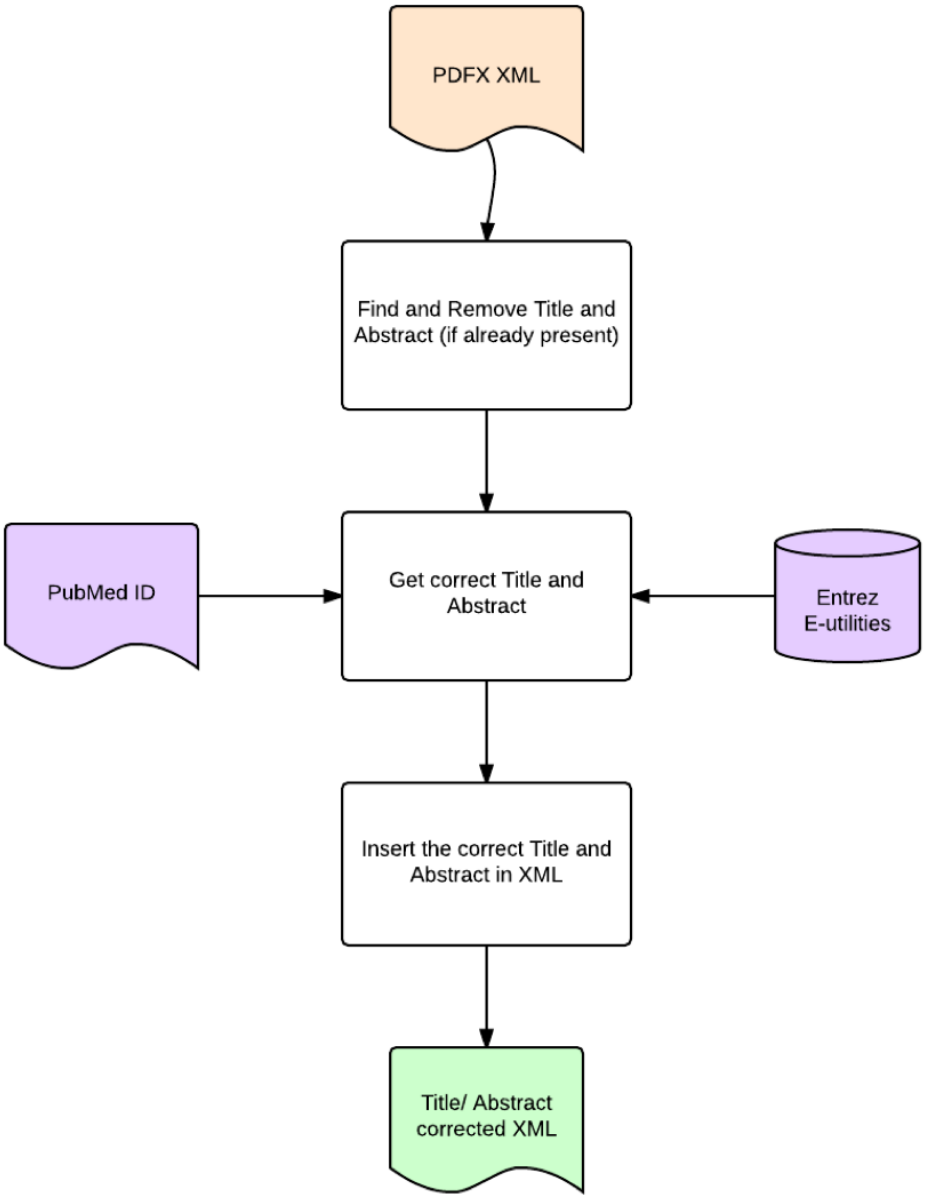
2.1 Title and Abstract Correction

The flow diagram illustrating this step is given in Figure 3. As the input of this step, PDFX performs well in detecting title and abstract for regular research articles; but in case of manuscripts which might have contents other than title or abstract on their first page, PDFX struggles to identify them correctly. To overcome this, we used the PMC Entrez e-utilities API (Sayers et al. 2011) to retrieve the correct title and abstract (using PubMed ID of the paper) and substituted them for whatever was present in PDFX XML. It should be noted that we applied PMC Entrez e-utilities API because our focus is the biomedical literatures archived in PubMed Central.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

229 **Figure 3. A flow diagram showing the title and abstract correction.**

230 The title and abstract correction pipeline, with PDFX XML as input and Title/Abstract corrected
231 XML as output.



232

233

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

2.2 Reference and Acknowledgement Correction

The format of references varies considerably among different publications ranging from a separate section towards the end to individual references in footnotes continuing alongside the content body. As PDFX is a purely rule-based system, it often falters in recognizing references when the common pattern of having references in the end with a heading is not present. Similar problems were noticed in identifying acknowledgements as well.

A key observation was that whenever PDFX identifies references/acknowledgement in a paper, it is generally correct and correction was needed only in the case when it does not identify either. We realized that the text content in references and acknowledgment showed clear patterns in language, and because for our purpose, we did not need to extract individual bibliographic items in the references section like PDFX does, classifying each section as a reference/acknowledgment was enough. Also, it is important to note that the only requirement was to detect reference text, which is much coarser grained and simpler than complete metadata extraction from journals, or metadata extraction for each citation.

Although references and acknowledgment are usually clumped together, in this correction step we designed two separate classifiers, as the patterns they show are different:

(1) For reference detection, we used PDFX XMLs which had a PDFX "reference" tag to train the classifier (there are 451 such XMLs in our experiment), by taking reference text as positive examples and rest of the text from the paper as negative examples. Some of the most prominent features were density of year like numbers, density of "et. al" and similar strings, density of numbers followed by the dot and so on. For each of the above features, the *ratio of the numbers found* versus *total number of tokens in the text* was used instead of the raw numbers. We

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

normalized these features and train a Random Forest classifier (Breiman 2001) to predict whether a paragraph is a reference text.

(2) For acknowledgment detection, we used PDFX XMLs which had a PDFX "acknowledgment" tag to train the reference classifier (there are 451 such XMLs in our experiment). We exploited another Random Forest classifier with features being TF-IDF followed by Latent Semantic Analysis (Deerwester et al. 1990) (Truncated Singular Value Decomposition (Hansen 1987; Kolda & O'leary 1998) in our experiment) to perform the detection. The features we adopted are TF-IDF of the phrases such as "funded by" and "express gratitude".

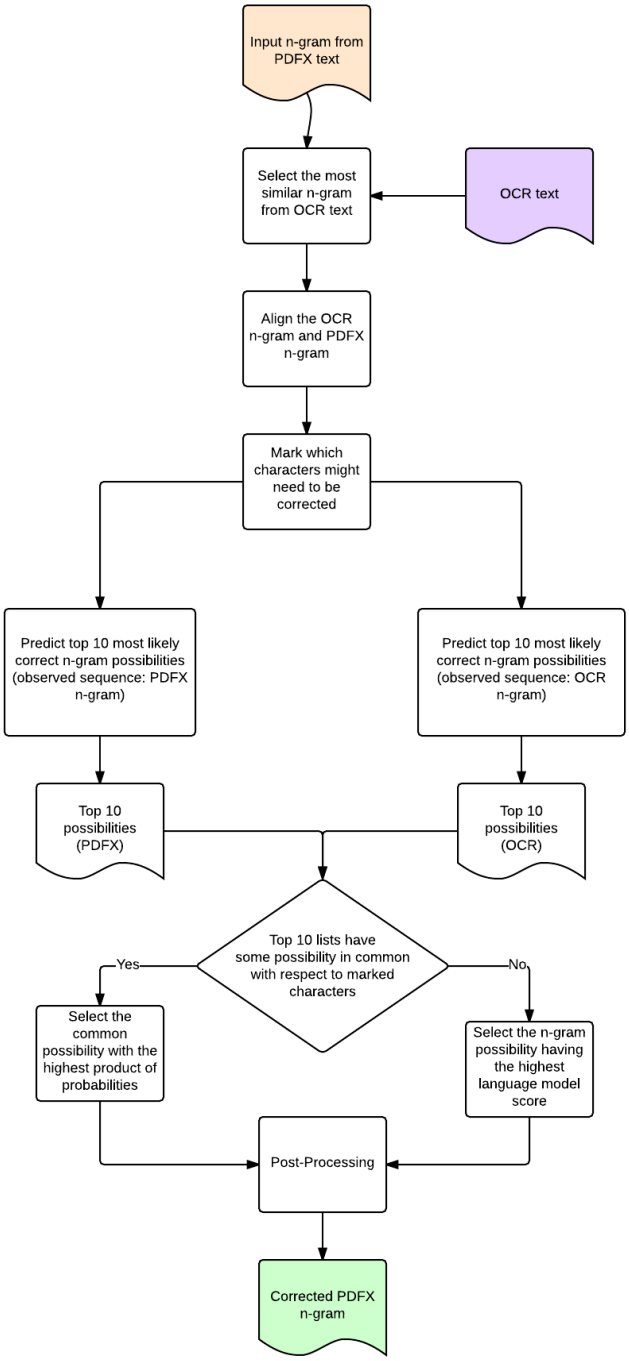
2.3 Special Characters Correction

PDFX and other PDF-to-text converters usually suffer from the problem of recognizing the special characters (such as '!' and '@') depending on the publishing of the PDF files. The basic idea for the special character correction step is that given a possibly erroneous n-gram from PDFX in the token/word level, we recognized the corresponding n-gram in Tesseract OCR and then aligned the two n-grams by their characters. Then, for differing characters we make a decision about what character should be present in that position using Hidden Markov Model (HMM) (Baum & Petrie 1966) with Viterbi inference algorithm (Viterbi 1967) and language models. The flowchart for this pipeline is summarized in Figure 4.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

Figure 4. A flow diagram for special character correction.

This pipeline shows the steps taken in correcting special character errors in an n-gram.



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

2.3.1 Suspicious N-Gram and Character Identification

Before applying HMM, we first select similar character-level n-grams (we treat each character, including the space, as unigram) from the output of OCR and PDFX, align the n-grams and then mark suspicious characters in the n-grams. To select the most similar n-gram from OCR and PDFX text, we used Levenshtein Distance (Levenshtein 1966) normalized for length to measure the similarity between the two n-grams. The alignment of the two n-grams is done at the character level and is based on a gene sequence alignment mechanism which is widely used (Hsu et al. 2008). If the n-gram in OCR text is at least 0.6 (in Levenshtein Distance) as like the PDFX n-gram then the characters that need to be replaced are marked. This is done by looking at the characters that differ in the two n-grams after the alignment and selecting those characters which are not aligned with whitespaces or digits. These *mismatch characters* are then fed to a HMM to predict top possible correct character. The example in Figure 5 show the OCR, PDFX, and gold standard n-grams. In this example, the mismatch characters in OCR n-gram are "α", "u", and "3", while that in PDFX n-gram are "a", "a" and "s".

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

294 **Figure 5. An example showing three types of n-gram: OCR, PDFX, and gold standard.**

295 The mismatch characters are shown in bold and underlined style.

1. OCR n-gram: "the ***α***-val**u**e wa**3** less than 0.01".
2. PDFX n-gram: "the **a**-val**a**e wa**s** less than 0.01".
3. Gold standard n-gram: "the **p**-val**u**e wa**s** less than 0.01".

296

297

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

2.3.2 Prediction of Top-N Possible Correct N-Gram Lists for OCR and PDFX

After marking the suspicious characters in the n-gram from OCR and the one from PDFX, we predict the list of the most probable correct n-grams for OCR and the list for PDFX. Our solution is to train an HMM (16) using a training corpus of aligned OCR, PDFX and gold standard texts, and infer the "most probable" n-gram by the Viterbi algorithm.

While formulating this problem in terms of HMM Viterbi inference, one of our assumptions is that characters in the words of meaningful sentences follow the Markov assumption to a good extent, something which is already well proven by handwriting and speech recognition applications, which have used HMMs to achieve good performance (10) (19). The other assumption is that OCR and PDFX have a statistical pattern or bias in the character level errors that they make, and that these patterns can be learned through a training corpus. The same technique of modeling words as sequences of characters by HMM has been used to boost OCR accuracy in (24).

To formally state the problem, we first define a character x_t as the t-th letter in an OCR n-gram. Therefore, each OCR n-gram can be represented as $X = x1x_2...x_N$, which is a concatenation of N characters. It should be noted that the character whitespace (i.e., " ") is also treated as a character. In Figure 5, the number of characters in the OCR n-gram "the α-value wa3 less than 0.01" is 30.

Next, we compute an *ambiguity dictionary*, which is a dictionary that records occurrences of observed mismatched characters in the corpus. Specifically, the ambiguity dictionary is a list of tuples of the form: (observed character, actual character, frequency). For example, a tuple ("α", "p", 30) means that for 30 times when we saw "α" in OCR / PDFX text corpus, the actual character in gold standard was "p". Multiple tuples for an observed character, like "α" in our example can be aggregated and written as:

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

320

321 "α": [("p", 150), ("α",30), ("a", 5)],

322

323 which means that for "α" in OCR / PDFX n-gram, the true character should be a "p" 150 times,
324 "α" 30 times and "a", 5 times.

325 Thus, each mismatch character (from OCR n-gram) x_t corresponds to a candidate-character set
326 δ_t, which can be computed from the ambiguity dictionary. For non-mismatch characters, the
327 candidate character set $\delta_t = \{x_t\}$, contains only one element which is exactly x_t. In our OCR n-
328 gram example, the candidate-token set of the first character "t" is "t", while that of the fifth
329 character "α" may be {"α", "a", "p", "o", "0"}, for example. Therefore, our problem can be
330 regarded as to choose the "most probable" candidate character for each mismatch character in
331 OCR n-gram, which can maximize the overall likelihood to recover the gold standard n-gram.
332 In other words, we want to find $Y^* = y_1 * y_2 * ... y_N = argmax_Y P(Y \mid X)$, where $y_1 * y_2 * ... y_N$ are
333 the "most probable" candidate characters given X, the pair of OCR and PDFX n-grams. We
334 simplify the notation by using P(Y) as the target probability that we want to maximize.

335 We apply Hidden Markov Model (HMM) to solve our problem. Based on the Markov
336 assumption, the probability we want to maximize can be written as $P(Y) = P(y_1 y_2 ... y_N) =$
337 $P(y_1)P(y_2 \mid y_1)P(y_3 \mid y_2)...P(y_N \mid y_N - 1)$. The model of our HMM is defined as follows:

338

339 • Observation

340 $\sigma = p_1, p_2, ..., p_M$, which is a set of all possible M values for all characters.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

- State

$$A = \bigcup_{t=1}^{N} \delta_t = \{q_1, q_2, \ldots q_K\}$$

which is a set of all possible K values for all candidate characters.

- Initialization Probabilities

$\Pi = \{\pi_i \mid \pi_i = P(y_t = q_i)\}$, which is a set of initial probabilities for each state. Therefore, $|\Pi| = K$. Suppose $L^1(q_i)$ is the conditional probability given by 1-gram language model (trained from the gold standard n-grams) for each candidate-character q_i (e.g., "α"), we compute π_i as follows:

$$\pi_i = L^1(q_i)$$

Note that

$$\sum_{i=1}^{K} \pi_i = 1$$

- Transition Probabilities

$A = \{a_{ij} \mid a_{ij} = P(y_{t+1} = q_j \mid y_t = q_i)\}$, which is a set of probabilities transiting from the t-th candidate character y_t with state q_i, to the (t + 1)-th candidate-character y_{t+1} with state q_j. Therefore, $|A| = K^2$. Suppose $L^2(q_i, q_j)$ is the conditional probability

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

given by 2-gram language model (trained from the gold standard n-grams) for two consecutive candidate-characters q_i and q_j (for example, "α-"), we compute a_{ij} as follows:

$$a_{ij} = L^2(q_i, q_j)$$

Note that

$$\sum_{j=1}^{K} a_{ij} = 1 \text{ for all states i = 1, 2, ..., K}$$

- Emission Probabilities

B = $\{b_{ik} | b_{ik} = P(x_t = p_k | y_t = q_i)\}$, which is a set of probability for the t-th candidate-character y_t with state q_i, to emit the t-th character x_t with observation p_k.Therefore, $|B|$ = K*M. Suppose $C(p_k, q_i)$ is the count that a candidate-character q_i (e.g., "p" from gold standard n-gram) is recognized as a character p_k (e.g., "α" from OCR n-gram), and λ is a regularization constant, we compute b_{ik} as follows:

$$b_{ik} = \frac{C(p_k, q_i) + \lambda}{\sum_{k'}^{M} (p_{k'}, q_i) + \lambda}$$

Note that, for all states i =1, 2, ..., K. In our example ambiguity dictionary for character "α" = [("p", 150), ("α", 30), ("a", 5)], the emission probability from "p" to "α" (i.e., the probability that we observe "α" in OCR n-gram, but the true character in gold standard n-gram is "p") can be estimated as ((30 + λ) / (150 + 30 + 5 + λ)).

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

Finally, we apply the Viterbi algorithm with our HMM model to decode the N most probable Y^*. In this way, we have the top N most probable n-grams given an OCR n-gram as the observation.

We follow the same process for PDFX as well, that is, treating the PDFX n-gram as the observation and using OCR n-gram to see which characters are not in consensus. Similarly, the ambiguity dictionary for PDFX was derived by applying the same method, except that we record ambiguities for PDFX instead of OCR. Hence, we can compute another list of the top N most probable PDFX n-grams along with their probabilities.

2.3.3 Combination of Top-N Possible Correct N-Grams Lists

Next, we select the most probable n-gram as the n-gram common in both lists (with respect to marked characters only) which has the maximum combined probability, using the idea of integrating gene mention tagging models (9). This is illustrated in Figure 6.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

Figure 6. The example showing two top-N list of the most probable NXML n-grams with their log probabilities given by Viterbi inference.

Candidate number 5 in the left list (where PDFX n-gram is the observation) matches candidate number 0 in the right list (where OCR n-gram is the observation), which is the "chosen best".

```
 0. -97.107 (rs2252931 max P = 2.2610 29   |   -87.240 (rs2252931 max P=2.2×10−9
 1. -97.410 (rs2252931 max P = 2.2×10 29   |   -90.024 (rsχ252931 max P=2.2×10−9
 2. -98.379 (rs2252931 maχ P = 2.2610 29   |   -90.306 (rs2252931 max P=2.2×10−9
 3. -98.681 (rs2252931 maχ P = 2.2×10 29   |   -90.312 (rs2252931 max P=2.2×10 9
 4. -98.901 (rs2252931 max P = 2.2610 −9   |   -90.507 (rs2252931 max P=2.2×10−9
 5. -99.204 (rs2252931 max P = 2.2×10 −9   |   -90.974 (rs2252931 Max P=2.2×10−9
 6. -99.725 (rs2252931 max P = 2>2610 29   |   -91.043 (rs2252931 maχ P=2.2×10−9
 7. -99.741 (rs2252931 max P 9 2.2610 29   |   -91.903 (rs2252931 max P=2.2×10−9
 8. -99.782 (rs2252931 Max P = 2.2610 29   |   -92.031 (rs2252931 Jax P=2.2×10−9
 9. -99.814 (rs2252931 max P 5 2.2610 29   |   -92.061 (rs2252931 max P=2.2×10−9

***Chosen best:
 0. -186.444400  -- (rs2252931 max P = 2.2×10 −9
```


PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

399 The PDFX n-gram in Figure 6 is "rs2252931 max P = 2.2610 29" and the OCR n-gram is:
400 "rs2252931 max P=2.2x10-9". The characters highlighted in green/blue are the characters marked
401 for correction. We won't apply correction to other characters, but instead just keep whatever we
402 observe in the PDFX n-gram because other mismatches are pairs of digits. The left list of 10
403 candidates is from treating the PDFX n-gram as the observation, while the right side shows the
404 candidates from treating the OCR n-gram as the observation. The numbers in negative are the log
405 probabilities for each sequence as given by the Viterbi algorithm.

406 The candidate numbered 5 in the left list matches the candidate numbered 0 in the right list
407 (for the highlighted characters), and this combination has the highest joint probability (although
408 other combinations also match like 5, 1 and 5, 4). Thus, we choose this candidate as the most
409 probable n-gram.

410 There can be a case where the two lists do not share a common candidate with respect to
411 marked characters. In that case, we use the n-gram language model trained on our corpus to
412 obtain the probabilities for each of these twenty candidates, and select the most probable n-gram
413 candidate. In case there is again a tie among all twenty, we take the top n-gram candidate from
414 the PDFX candidate list as the most probable n-gram. An example of this scenario is shown in
415 Figure 7, where the two lists do not share a candidate in common and the language model chooses
416 the best candidate in the OCR candidate list, as it scores the highest.

417 Finally, in the post-processing step, we pick th n-gram with the highest score, as the "most
418 probable" n-gram as the output of the special character correction step.

419

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

Figure 7. The example showing two top-N list of the most probable n-grams with their log probabilities given by Viterbi inference.

None of the pair of candidates in the lists match.

```
 0. -76.754 P = 9.0610 29 Beta 21.9  |  -75.916 P= 9.0×10−9 Beta −1.9
 1. -78.122 P = 9.0×10 29 Beta 21.9  |  -76.056 P= 9.0×10−9 Beta −1.9
 2. -78.548 P = 9.0610 −9 Beta 21.9  |  -76.194 P= 9.0×10−9 Beta −1.9
 3. -78.716 P = 9.0610 29 Beta −1.9  |  -77.299 P= 9.0X10−9 Beta −1.9
 4. -79.388 P 9 9.0610 29 Beta 21.9  |  -77.439 P= 9.0X10−9 Beta −1.9
 5. -79.460 P 5 9.0610 29 Beta 21.9  |  -77.577 P= 9.0X10−9 Beta −1.9
 6. -79.583 P 4 9.0610 29 Beta 21.9  |  -78.462 P= 9.0×10−9 Beta +1.9
 7. -79.710 P 3 9.0610 29 Beta 21.9  |  -78.582 P= 9.0−10−9 Beta −1.9
 8. -79.794 P 2 9.0610 29 Beta 21.9  |  -78.659 P= 9.0×10 9 Beta −1.9
 9. -79.916 P = 9.0×10 −9 Beta 21.9  |  -78.723 P= 9.0−10−9 Beta −1.9

***Chosen best:
Nothing in common in the nbest lists...
Lang model score OCR top:  -23.6266403198 P = 9.0×10 −9 Beta −1.9
Lang model score PDFX top:  -25.0832328796 P = 9.0610 29 Beta 21.9
```


PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

2.4 Jumbling Correction

The correction of jumbling errors (the wrong order of text while converting PDF to text, as illustrated in Figure 1) in the output XML from PDFX is done by using the textual output as generated by Apache PDFBox. We start by making a non-overlapping n-gram set of PDFX XML text and an overlapping n-gram set for Apache PDFBox text. Then, we use a simple exhaustive method to check if a n-gram contains jumbling error. That is, if an overlapping n-gram is a permutation of another n-gram, we consider that the n-gram contains jumbling error. Next, we replace the suspicious n-gram by the non-jumbled version of n-gram from Apache PDFBox. Finally, we string together the non-overlapping n-grams with a space to construct the jumbling corrected text.

2.5 Word Boundary Correction

To correct word boundary error (e.g., “cancer” might become “can-” and “cer” if it is the last word of a row, thus it should be combined and corrected as “cancer” instead of two words), we adopted a simple dictionary lookup method by comparing different parts (e.g., “can-” and “cer”) with the English dictionary in the Enchant Spellchecking System (2015c) and merging them as required. Specifically, if the hyphenated word is not present in the English dictionary but the word without the hyphen is present in the dictionary, we remove the hyphen from the word. If the word with or without hyphen is not present in the dictionary, we keep it unchanged.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

2.6 Experiment Settings

We conduct experiments to compare bioPDFX with state-of-the-art PDFX tool (Constantin et al. 2013), and test the following two hypotheses:

(a) For each of the individual correction steps, bioPDFX provides better quality of PDF to XML conversion.

(b) For overall pipeline, bioPDFX also provides better conversion quality.

In our experiment, we set n=5 for n-grams. For special character correction, we exploit StochHMM (14) to calculate the top-N possible correct n-Grams lists by Viterbi inference, and KenLM (8) to build language models. We set the regularization parameter to 10^{-6}.

2.6.1 Dataset and Gold Standards

To test the two hypotheses, National Human Genome Research Institute (NHGRI) provided 2,185 biomedical literatures (Jain et al. 2016) related to Genome Wide Association Study (GWAS) (Hindorff et al. 2009; Welter et al. 2014). Among them, 941 are indexed by PubMed Central (PMC), where both PDF and XML versions are available for download as we searched in 2015. From these articles, we randomly sampled 100 pairs of PDF and XML versions (i.e., NXMLs) from these 941 articles as gold standards for four of our correction tasks: title and abstract, special characters, jumbling, and word boundary. For reference and acknowledgement, we built our own gold standard from PDF files directly rather than using those present in NXMLs. The reason behind

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

this was incomplete reference sections in most of NXMLs and complete omission of acknowledgement sections. For 70 randomly selected PubMed articles, we manually compiled references and acknowledgement text from their PDFs to act as gold standard. For hypothesis (b), we further manually labeled the p-value and numbers (which are import for GWAS literatures) as our gold standard. We randomly split the data in to 50% training and 50% test.

2.6.2 Evaluation Metrics

We developed a scoring metric which measured how a corrected XML from our system performs with respect to the corresponding NXML. In this metric, an n-gram based similarity method is used which is common in plagiarism detection. The idea is to make two sets of n-grams from NXMLs and from our system, and use them to measure the occurrence and correct order of words. We then compute the F1-score of the two n-gram sets as our evaluation metrics for title and abstract, reference and acknowledgement, special characters, and word boundary. We applied macro F1-score for most of the correction tasks except special characters and word boundary (of which micro F1-score is computed). This is because the number of special characters or words with incorrect boundary in an article can vary considerably. For jumbling error, we cannot apply the n-gram similarity methods directly, as the error may appear in the tables (as shown in Figure 1). Therefore, we compute the average counts of such errors over all the articles in the dataset as our evaluation metrics. For hypothesis (b), we further compute the micro F1-score for the p-value and numbers (for the same reason as special character and word boundary), and the macro F1-score for the overall text quality (which is the n-gram similarity score for the whole document).

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

3. Results

The results for each of the five correction steps are shown in Table 1. That is, each step is evaluated independently. As Table 1 depicts, in all correction steps bioPDFX outperforms PDFX, especially for abstract, reference/acknowledgement, special character, and jumbling error corrections.

On the other hand, Table 2 shows the scores at the end of the pipeline, where the corrections are run in order of the pipeline as shown in Figure 2. Like the results of individual corrections steps, bioPDFX in general perform better. The additional evaluation tasks (p-value, number and overall text quality) are also included in Table 2. It should be noted that although no correction is aimed at improving p-values, numbers or overall text quality, bioPDFX was still able to provide higher conversion quality for these three tasks.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

501 **Table 1. Stage wise quality score comparison of raw and corrected XMLs.**

Correction Step	PDFX	bioPDFX	Metric
Title	0.9135	0.9763	Macro F1
Abstract	0.5428	0.8920	Macro F1
Reference/Acknowledgment	0.6496	0.8026	Macro F1
Special Character	0.7071	0.8860	Micro F1
Jumbling	2.89	1.92	Average Error
Word-Boundary	0.7619	0.8053	Micro F1

502 Jumbling score is the average number of jumbling errors detected per paper (lower is better).

503

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

504 **Table 2. Final quality score comparison of raw and corrected XMLs at the end of the**
505 **correction pipeline.**

Correction Step	PDFX	bioPDFX	Metric
Title	0.9135	0.9695	Macro F1
Abstract	0.5428	0.8877	Macro F1
Reference/Acknowledgment	0.6496	0.8026	Macro F1
Special Character	0.7071	0.8932	Micro F1
Jumbling	2.89	1.95	Average Error
Word-Boundary	0.7619	0.8189	Micro F1
P-value	0.5465	0.6615	Micro F1
Number	0.8733	0.8853	Micro F1
Overall Text Quality	0.9006	0.9107	Micro F1

506 Jumbling score is the average number of jumbling errors detected per paper (lower is better).

507

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

4. Discussion

Based on the experiment results, we show the experimental evidence to support the two hypotheses. Next, we discuss our results for each correction step in detail:

- *Title and Abstract*. The correction scores for title and abstract is quite good in case of raw XMLs and increases substantially in the final corrected XMLs. This is an expected behavior as we are retrieving titles and abstracts using the PMC Entrez e-utilities API directly which can be thought of as a gold standard itself. The F1-scores are still not perfect due to syntactic differences between title/abstract text present in NXMLs and the title/abstract text retrieved from e-utilities API. For example, the API does not seem to use Unicode characters like "≤", "≥" and "~" and would instead have the phrases "less than or equal", "greater than or equal" and "approximately" in their place. Furthermore, there were cases where the data from NXMLs was incorrect because of errors from authors of those papers like missing title/abstracts and other similar inconsistencies in punctuation. Apart from this, we can consider the titles and abstracts to be near perfect in the corrected XMLs (assuming e-utilities API gives correct results). Some other errors are introduced due to accumulation of errors from previous stages of corrections in the pipeline, but as can be seen by comparing stage-wise and end of pipeline scores in Table 1 and Table 2, they are quite small and can be neglected.

- *Reference and Acknowledgement*. Reference/Acknowledgment scores also improved considerably and lead to better scores for the remaining stages of the pipeline, by removing a significant number of the False Positives, as can be observed by comparing the stage-

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

wise-results in Table 1 and final-results in Table 2.

- *Special Character*. We could improve special character F1-score by a considerable margin through our technique of combining HMMs, n-gram alignment, OCR, and language models.

- *Jumbling Error*. The average counts of jumbling errors are fewer for bioPDFX then for PDFX. It should be noted that there is no guarantee that jumbling errors occur only over 5-grams, or equivalently within 5 words. In the same document, there might be jumbling errors spanning up to arbitrarily large and varied n. A better approach might be to start with a much larger n and attempt to detect jumbling from there and progressively reduce n one by one, however this approach is computationally intensive. Even operating on 5-grams for jumbling error correction step takes a significant amount of time compared to other correction steps.

- *Word Boundary*. The F1-score of word boundary correction using bioPDFX are also improved modestly comparing to PDFX. We also attempted to use a language model (along with the English dictionary) to increase the range of corrections possible for biomedical terms, but it did not give us any performance improvement. We believe this is because word boundary errors are comparatively infrequent compared to total number of words in an article, and because most words in an article are English words or numbers, most word boundary errors happen in English words, which are already corrected by the English dictionary.

Also, the usability of bioPDFX can be largely increased by a multiple-files uploading functionality, so that users can submit many PDF files of interest for conversion instead of

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

552 uploading one-by-one. We are currently implementing an interface for such a multiple-uploading
553 feature (Figure 8) for a biocuration tool that will include this feature to maximize the utility of
554 bioPDFX.

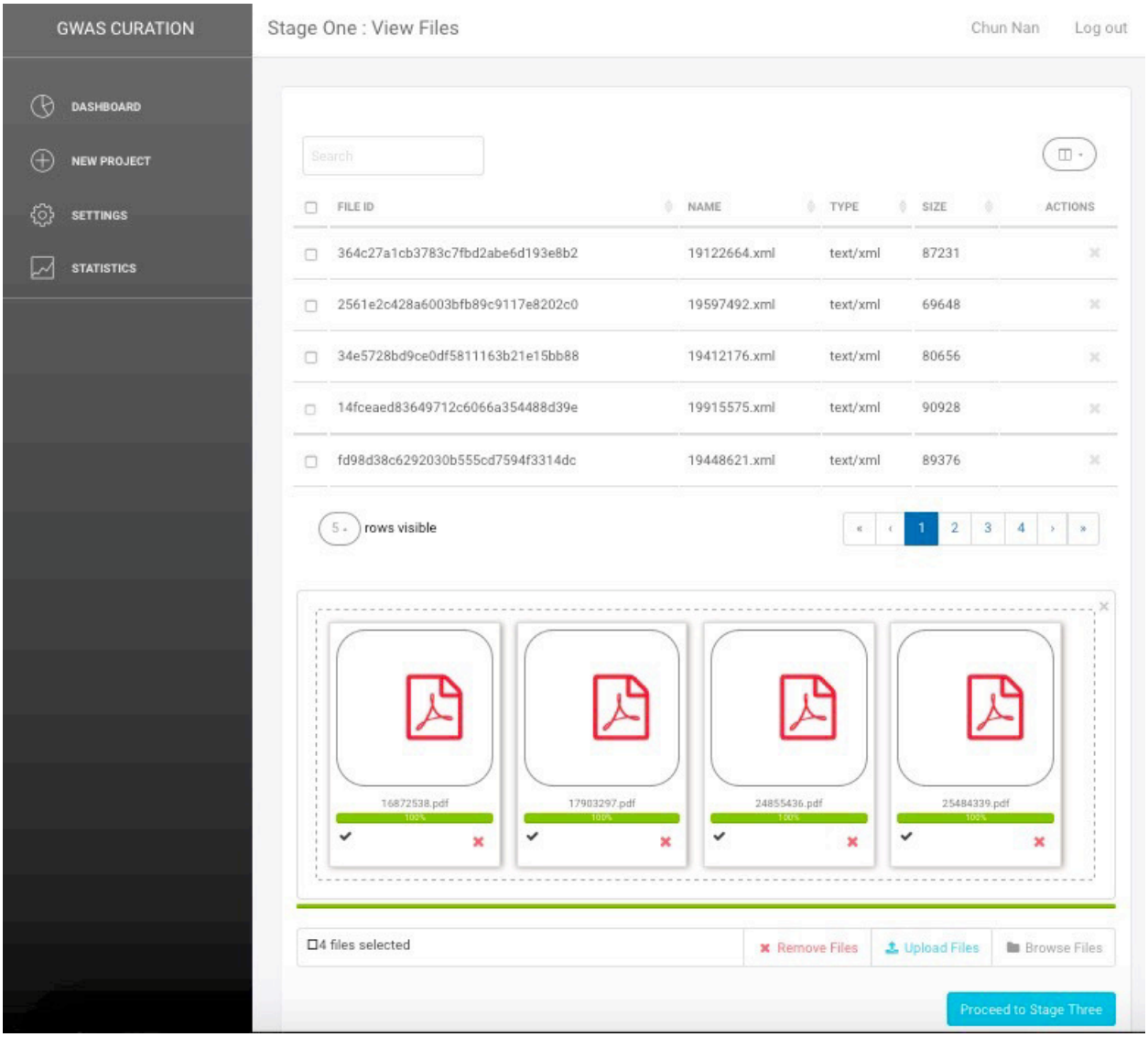
555

556

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

Figure 8. The interface for multiple-files uploading.

The usability of bioPDFX can be largely increased by integrating it within a biocuration tool.



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

5. Conclusions

We have developed a tool to convert PDF files into XML format specifically for biomedical articles, targeting the five challenges of state-of-the-art converters: title/abstract, reference/acknowledgement, special character, jumbling error, and word boundary. Overall, the whole pipeline developed in this paper makes the published literature in form of PDF files much better suited for text mining tasks while slightly improving the overall text quality as well. Although the tool discussed in this paper is trained to be specifically optimized for our target corpus (GWAS), we believe that these techniques can be applied to other kinds of literatures as well.

It should be noted that although we use PDFX XMLs as the main text sources, and use Apache PDFBox and OCR text as auxiliary text sources, bioPDFX is general and is not dependent on these text sources. Given a primary text source to correct substitution errors, other secondary text sources can also be exploited in the correction steps.

There are some limitations of the approach, but we believe that these can be easily overcome in different domains and by investigation of several different text sources. These limitations are summarized as below:

- Title and abstract corrections depend on the PMC Entrez e-Utilities API
- As we model n-grams as sequences of characters and due to the Markov assumption, we cannot correct errors in which one character is replaced by multiple ones, or is deleted
- Jumbling error correction is only an approximation since we fix n=5

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

583 In the future, we plan to run each of the text sources in parallel further improving the overall
584 performance, for which our initial result was reported in (Goyal et al. 2016). Also, we plan to
585 explore more challenging issues in the PDF to XML conversion process. Finally, we plan to extend
586 our experiments on more diverse types of large-scale corpuses.

587

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

Acknowledgements

We would like to thank Dr. Lucia Hindroff of NHGRI for her generous support of this project by providing us PDF copies of test articles and the curation guidelines of the Catalog of GWAS to make this research possible. We also would like to thank Dr. Helen Parkinson, Dr. Jackie MacArthur and their team members at EBI for their assistance.

References

2015a. Adobe Acrobat SDK. *Available at* http://www.adobe.com/devnet/acrobat/overview.html.

2015b. Apache PDFBox. *Available at* https://pdfbox.apache.org/index.html.

2015c. Enchant Spellchecking System.

2015d. Mozilla PDF.js. *Available at* https://mozilla.github.io/pdf.js/.

2015e. National Center for Biotechnology Information (NCBI), National Library of Medicine (NLM), PubMed Central (PMC) Open Access Subset. *Available at* http://www.ncbi.nlm.nih.gov/pmc/tools/ftp/.

Baum LE, and Petrie T. 1966. Statistical inference for probabilistic functions of finite state Markov chains. *The annals of mathematical statistics* 37:1554-1563.

Berg ØR. 2011. High precision text extraction from PDF documents.

Breiman L. 2001. Random forests. *Machine Learning* 45:5-32.

Constantin A, Pettifer S, and Voronkov A. 2013. PDFX: fully-automated PDF-to-XML conversion of scientific literature. Proceedings of the 2013 ACM symposium on Document engineering. Florence, Italy: ACM. p 177-180.

Cortez E, da Silva AS, Gonçalves MA, Mesquita F, and de Moura ES. 2007. FLUX-CIM: flexible unsupervised extraction of citation metadata. Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries: ACM. p 215-224.

Deerwester S, Dumais ST, Furnas GW, Landauer TK, and Harshman R. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41:391-407. 10.1002/(sici)1097-4571(199009)41:6<391::aid-asi1>3.0.co;2-9

Garcia A, Murray-Rust P, Burns G, Stevens R, Tkaczyk D, McLaughlin C, Belin A, Di Iorio A, García L, and Gruson-Daniel C. 2013. PDFJailbreak-a communal architecture for making biomedical PDFs semantic. *Proceedings of BioLINK SIG 2013*:63.

Goyal A, Singh A, Bhargava S, Crawl D, Altintas I, and Hsu C-N. 2016. Natural Language Processing using Kepler Workflow System: First Steps. *Procedia Computer Science* 80:712-721.

Hansen PC. 1987. The truncatedsvd as a method for regularization. *BIT Numerical Mathematics* 27:534-553.

Hindorff L, Sethupathy P, Junkins H, Ramos E, Mehta J, Collins F, and Manolio T. 2009. Potential etiologic and functional implications of genome-wide association loci for human diseases and traits. *Proceedings of the National Academy of Sciences* 106:9362-9367. citeulike-article-id:4806365
doi: 10.1073/pnas.0903103106

Hsu C-N, Chang Y-M, Kuo C-J, Lin Y-S, Huang H-S, and Chung IF. 2008. Integrating high dimensional bi-directional parsing models for gene mention tagging. *Bioinformatics* 24:i286-i294. citeulike-article-id:12926636

Jain S, Tumkur K, Kuo T-T, Bhargava S, Lin G, and Hsu C-N. 2016. Weakly Supervised Learning of Biomedical Information Extraction from Curated Data. *BMC Bioinformatics*.

Kolda TG, and O'leary DP. 1998. A semidiscrete matrix decomposition for latent semantic indexing information retrieval. *ACM Transactions on Information Systems (TOIS)* 16:322-346.

Levenshtein VI. 1966. Binary codes capable of correcting deletions, insertions and reversals. Soviet physics doklady. p 707.

Ramakrishnan C, Patnia A, Hovy E, and Burns GA. 2012. Layout-aware text extraction from full-text PDF of scientific articles. *Source code for biology and medicine* 7:1.

Sayers EW, Barrett T, Benson DA, Bolton E, Bryant SH, Canese K, Chetvernin V, Church DM, DiCuccio M, and Federhen S. 2011. Database resources of the national center for biotechnology information. *Nucleic acids research* 39:D38-D51.

Smith R. 2007. An overview of the Tesseract OCR engine.

Tkaczyk D, Szostek P, Dendek PJ, Fedoryszak M, and Bolikowski L. 2014. CERMINE--Automatic Extraction of Metadata and References from Scientific Literature. Document Analysis Systems

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017

(DAS), 2014 11th IAPR International Workshop on: IEEE. p 217-221.
Tong X, and Evans DA. 1996. A statistical approach to automatic OCR error correction in context. Proceedings of the fourth workshop on very large corpora. p 88-100.
Velagapudi P. 1999. Using hmms to boost accuracy in optical character recognition. Proceedings of SPIE, 27th AIPR Workshop: Advances in Computer-Assisted Recognition: Citeseer. p 96-104.
Viterbi A. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE TRANSACTIONS ON INFORMATION THEORY* 13:260-269.
Welter D, MacArthur J, Morales J, Burdett T, Hall P, Junkins H, Klemm A, Flicek P, Manolio T, Hindorff L, and Parkinson H. 2014. The NHGRI GWAS Catalog, a curated resource of SNP-trait associations. *Nucleic acids research* 42:D1001-D1006. citeulike-article-id:12826571
doi: 10.1093/nar/gkt1229
Zhuang L, and Zhu X. 2005. An OCR post-processing approach based on multi-knowledge. International Conference on Knowledge-Based and Intelligent Information and Engineering Systems: Springer. p 346-352.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2993v1 | CC BY 4.0 Open Access | rec: 26 May 2017, publ: 26 May 2017