ペーパーレスでの特許文献精読ツールの開発

審査第一部分析診断　　樋口 祐介

抄録

本記事では、筆者が開発したペーパーレスでの特許文献の精読を支援するツールについて、開発のきっかけや検討した事項、ツールの機能の紹介等を行う。

なお、本ツールは筆者が自分の業務のペーパーレス化・効率化のために個人的に開発したものであり、本記事に記載の内容は組織の方針や見解と何ら関係ないものである。

1. 開発のきっかけ

新型コロナウイルスの感染拡大により緊急事態宣言が初めて発令された2020年3月時点で筆者は調整課に在籍しており、審査室を支援する業務に携わっていた。

調整課でのテレワークにおいては、意思疎通に多少の工夫を要する程度でそこまで大きな障害を感じなかったが、同時に審査室でもテレワークが浸透していく様を見ていると、自分は審査室に戻ったときにテレワーク環境で審査をこなす自信がなかった。

一番不安に感じたのは、本願や発見した先行技術文献を印刷し、紙に手書きでメモを書きつつ精読するスタイルの脱却である。自宅に職場と同等の印刷環境を整えるのは難しく、何らか業務の方法を変える必要があることは明らかであった。

当初はペーパーレス審査で紙審査と同じパフォーマンスを出すのは困難だと思ったが、その理由について突き詰めて考えていくと、ペーパーレス審査が本質的に紙審査に劣っているわけではなく、単にこれまでの職場の環境が紙審査に適した形に最適化されているだけではないかとの考えに到った。

座席から手の届く位置に高速で印刷できるプリンタが配備されており、紙資料を保管する広い収納スペースが各審査官に与えられている。担当する案件の仮包袋や上げ書類が手元に届くようにシステム面・体制面が整備されており、審査済みの案件の仮包袋を収納するスライダックも各フロアに整備されている。このように紙審査を支援する環境が多く整っている状況では、ペーパーレス審査に不安を覚えるのも仕方がないかと思う。

そうであれば、ペーパーレス審査を行う上で必要なのは、それを支援する環境の整備である。長年の工夫により磨かれてきた紙審査の環境を一朝一夕で超えることは難しいが、ペーパーレス審査はデジタル技術を活かしたあらゆる可能性を秘めているという点で将来性があると考えられる。

筆者の業務は庁内のシステム開発と関係の薄いものであったが、ペーパーレス審査の支援ツールは、特にUI周りで好みが分かれそうであったことから、自分に最適化されたツールを自分で開発することにした[1)]。

2. 目標の設定

「ペーパーレスで特許文献を精読でき、且つ書き込みが可能なツールの開発」を目標として設定した。このようなツールがあればペーパーレス審査に

1) 関係部署の尽力により、本記事の執筆時点（2022年6月）でペーパーレス審査を支援する環境が相当程度整っている点を付言する。

問題なく移行できると思ったのが理由である。

2.1 特許文献の精読支援機能

特許審査における特許文献の精読には以下の特徴がある。

(1) 明細書に目次がなく、文章全体が構造化されていない。結果として、どこに何が書いてあるかを把握するのに時間を要する。

(2) 明細書全体を上から順に読むのではなく、「必要な情報」が書いてある箇所を虫食い的に読む。「必要な情報」は、本願の各請求項に対応する記載や、用語の定義に関する記載、検索式にヒットしたワードに関する記載などであり、一つの文献を精読している間に何度も変遷する。

これらの特徴から、特許文献の精読を支援する機能としては、「必要な情報」が書いてある箇所を効率的に特定できる高い検索性が求められる。

ペーパーレスの精読方法として、「文献をPDFファイルの形で出力して、そのPDFファイルにメモを書き込む」方式がすぐに思い浮かぶ。しかし、一般的なPDFリーダーの検索機能はCtrl+Fによる文字列検索程度であり、電子的に文字列を検索できるのは紙審査に比べて便利であるとはいえ、他の紙審査が有する利点を補うには不十分であると感じている。

よって、単純な文字列検索よりも検索性に優れる機能の開発を目標とした。

2.2 書き込み機能

紙審査において文献を精読する際には、ペンやマーカーを使って紙に書き込みを行うのが普通であるが、書き込みには主に二つの役割があると考えられる。

(1) 書き込みを残すことにより、後日同じ文献を読む際に内容を効率的に把握できる。

(2) 書き込みながら文献を精読することで、より深く文献を理解することができる[2)]。

特に(2)の役割が存在することをふまえると、文献の精読を支援するツールを開発するのであれば、書き込み機能も実装すべきという結論になる。

紙審査の優れた点としては書き込みの容易さが第一に挙げられることが多く、ツール開発にあたっては書き込み機能の使い勝手(UI/UX)も重要であると考えられる[3)]。

2.3 ペーパーレスならではの機能の追求

紙審査とペーパーレス審査にはそれぞれメリット・デメリットがあるが、紙審査の特性や環境をペーパーレス審査において忠実に再現することには現実的でない部分がある。それよりも、紙審査にはできなかった、デジタル技術を利用できるペーパーレス審査だからこそ可能なことを追求することで、結果的に紙審査より優れたペーパーレス審査の実現を目指す方が建設的である[4)]。

ツールの開発にあたっては、可能な範囲でペーパーレスならではの機能を検討し、実装することも目標の一つとした。

3. 開発したツールの紹介

3.1 全体像

ツールはHTMLの形式でWebブラウザ上で動作し、サーバー側の処理を必要としない(クライアント側の処理だけで完結する)ものとした。このような構成とすることで、サーバーの管理やインストール作業が不要であり、職場のPCでそのまま動作するツールを実現した。

PDFファイルには環境に依らず同一の見栄えを保持できる利点があるが、取り扱いが技術的に困難であることや、検索性に優れたツールを作成するうえでのフォーマットとしては適していないと判断したことから、文献の情報はHTMLの形で保持することとした。

---

2) “Most importantly, annotated content can help readers obtain a deeper and broader understanding compared to digital content without annotations.” (Chen, Chih-Ming, and Fang-Ya Chen. "Enhancing digital reading performance with a collaborative reading annotation system." Computers & Education 77 (2014) :67-81)

3) “First of all, the annotation action must be effortless in all senses - easy to access and visualize, as few interactions as possible and incontext interactions to minimize the lose focus.” (Kawase, Ricardo, Eelco Herder, and Wolfgang Nejdl. "A comparison of paper-based and online annotations in the workplace." European Conference on Technology Enhanced Learning. Springer, Berlin, Heidelberg, 2009.)

4) “From the above there is evidence that due to inherent differences when moving from the paper-based world to electronic devices, the character of annotations will necessarily change.” (脚注3と同一の論文より)

Webブラウザ上で書き込みを行った後、保存ボタンを押すと、文献のテキスト、図面の画像、書き込みの情報、CSSのコード、JavaScriptのコードが全て埋め込まれたHTMLファイルがダウンロードされる。ダウンロードしたファイルはWebブラウザで開けば状況が復元され、再度メモを行うことができる。また、一つのHTMLファイルに文献の内容やプログラムが全て含まれているので、好きなフォルダに保存でき、メールで送って他人に共有することもできる。

なお、開発当初はJavaScriptを直接記述していたが、ツールの規模が大きくなり、効率的に不具合の少ないツール開発を行う必要性が高まったため、途中から開発環境としてTypeScript + Node.js + webpackを採用し、記述したTypeScriptのコードをクライアント側で動作するJavaScriptにトランスパイルする方法に切り替えて開発を行った。

3.2 外観

ツールの外観を図1に示す。画面左側に図面、右側に本文（書誌事項、特許請求の範囲、明細書）が表示され、それぞれ独立にスクロールできる。

明細書に図面のタグ(【図1】など)がある場合には、対応する図面の上部にタグ内の文章が表示される。

3.3 主な機能

3.3.1 スペクトルバー

特許審査官にはお馴染みの、指定したワードの色を反転し、ワードの出現位置をスペクトルバーに表示する機能を実装した。普段の審査において先行技術調査により発見した文献を読む際には、スペクトルバーを補助的に用いることがほとんどであり、我々審査官は日々スペクトルバーを使って文献を効率的に精読するトレーニングを行っているようなものである。したがって、スペクトルバーが実装されたメモツールを開発すれば必ず良いものになるという思いを開発当初から持っていた。

既存のスクリーニングツールでも、ワード反転した結果を紙に印刷して読むことは可能だが、一度印刷してしまうと反転ワードを変更できない。2.1にて考察したように、一つの文献を精読するうえで「必要な情報」は何度も変遷していくものであり、その度に反転ワードも変更されるべきであるから、任意のタイミングで反転ワードを変更できるツール

図1

特許第6691280号を本ツールで表示した画面の例。画面左側から図面、本文、コメントエリア、スペクトルバーが表示される。スペクトルバーには、反転ワードの位置に加えてコメントやマーカーを付した位置が左側に表示される。

の開発が求められていると感じていた。

反転ワードの変更は頻繁に行われると想定し、複数のワードをスペース区切りで入力でき、反転色は自動で設定され、＋で区切ったワードはシソーラスとして同じ色で反転されるようにするなど、操作の手間を省く工夫を行った。図2は反転ワード設定画面の拡大図を表す。

スペクトルバーをクリックすると本文の対応する位置にジャンプする機能も実装しており、興味のあるワードが登場する箇所をすぐに表示できるようになっている。これも紙による精読では実現できないものであり、「必要な情報」を効率的に探索できると期待される。

また、反転されるワードをより自然な区切りへと拡張するブロック反転機能も実装した。図1において、ワードとして「特許文献」を指定しているが、「各特許文献」「当該各特許文献」といった区切りで反転されている点に注目いただきたい。ハイライトの区切りは自然である方が良いと主張する論文[5]を見つけたので実装してみたが、個人的には読みやすくなったと感じている。（ただし、単純なルールベースで区切り位置を拡張しているため、区切り位置が不自然な箇所もある。）

さらに、ワードとして正規表現も指定できるようにしており、例えば「/請求項\d+/」と入力すると「請求項○」（○は数字）のようなフレーズが反転されるなど、柔軟な反転が可能である。正規表現を適切に利用することで、反転の区切りがより自然なものとなる。

精読を終えてHTMLファイルをダウンロードする際には、HTMLファイル内に設定した反転ワードの情報が埋め込まれるようにしており、ファイルを再度開くと反転ワードも含めて画面が復元される。以前精読した際にどのようなワードに注目していたかを把握できるので、文献の再理解が効率化される。また、ファイルを再度開いたあとに別の反転ワードを設定することもでき、以前とは別の観点で精読する際に有用である。

3.3.2 マーカー、コメント

Microsoft Wordを参考に、選択したテキストをマーカーで塗り潰す機能と、右側に吹き出しの形でコメントを残す機能を実装した。筆者の経験上、文献に何かメモを書き込みたい欲求の大半はこの二つの機能により満たされる。

また、マーカーやコメントを付けた位置をスペクトルバー上に同色のマークで表示させるようにした。紙審査での付箋に相当する機能であり、メモを残した部分を一目で把握でき、その部分をクリックすることでメモを含めて表示することが可能である。本文の内容に加えて、自分が残したメモを素早く把握したいケースも多々あり、そのような場面に有効である。

3.3.3 本文の編集

画面中央に表示されている本文は、テキストエディタのような操作感で編集することが可能となっている[6]。無論、意図しない本文の改変は文献の認定誤りに繋がるため、デフォルトでは編集を許可しない状態になっているが、本文が編集可能なことで、例えば以下のような、紙審査にはできない使い方ができるようになる。

(1) 本文の適当な位置に改行を挿入し、本文を読みやすくする。特に、特許請求の範囲の文章は見

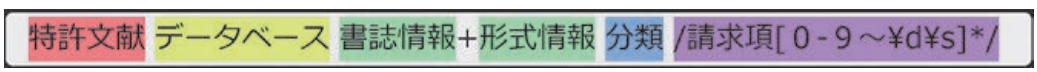

図2

反転ワード設定画面の拡大図。スペース区切りで複数のワードを入力してEnterキーを押下すると、自動で色を設定したうえでワード反転が行われる。また、＋で区切ったワードは同じ色で反転され、/（スラッシュ）で囲んだワードは正規表現として解釈される。

5) “We hypothesize that when text-highlighting in digital displays becomes more convenient and natural for the reader, then more cognitive resources will be available for deeper processing of the text and, consequently, better comprehension will be found.” (Ben-Yehudah, Gal, and Yoram Eshet-Alkalai. "The contribution of text-highlighting to comprehension: A comparison of print and digital reading." Journal of Educational Multimedia and Hypermedia 27.2 (2018) : 153-178)

6) 厳密には、HTMLにおいてcontenteditable属性を追加した要素となっている。

栄えを整えることで可読性が向上するケースが多い。

(2) 文の一部のテキストの書式を変更する。例えば、「実施例○」などの見出しに相当する文字を大きくすることで、本文の全体像を把握しやすくなる[7]。

(3) 本文中にメモを追加する。メモの量や内容によっては、前述のコメント機能よりも本文にメモを挿入する形式の方が読みやすいこともある。

3.3.4 図面符号分析

図面をダブルクリックするとサブウィンドウが起動し、図面内に存在する符号と対応するテキストの一覧が図面と並んで表示される（図3）。

図面内に存在する符号と対応するテキストは、図面の画像処理と明細書のテキスト分析により抽出している。誤りの無い抽出は困難だが、誤っている場合には簡単に手動で修正できる機能を実装しており、実用上はさほど問題にならないかと思っている。

紙審査においては、印刷した図面の符号に対応するテキストを調べ、場合によってはテキストを図面に書き込む作業に時間を要するが、本機能によりその負担が軽減される。

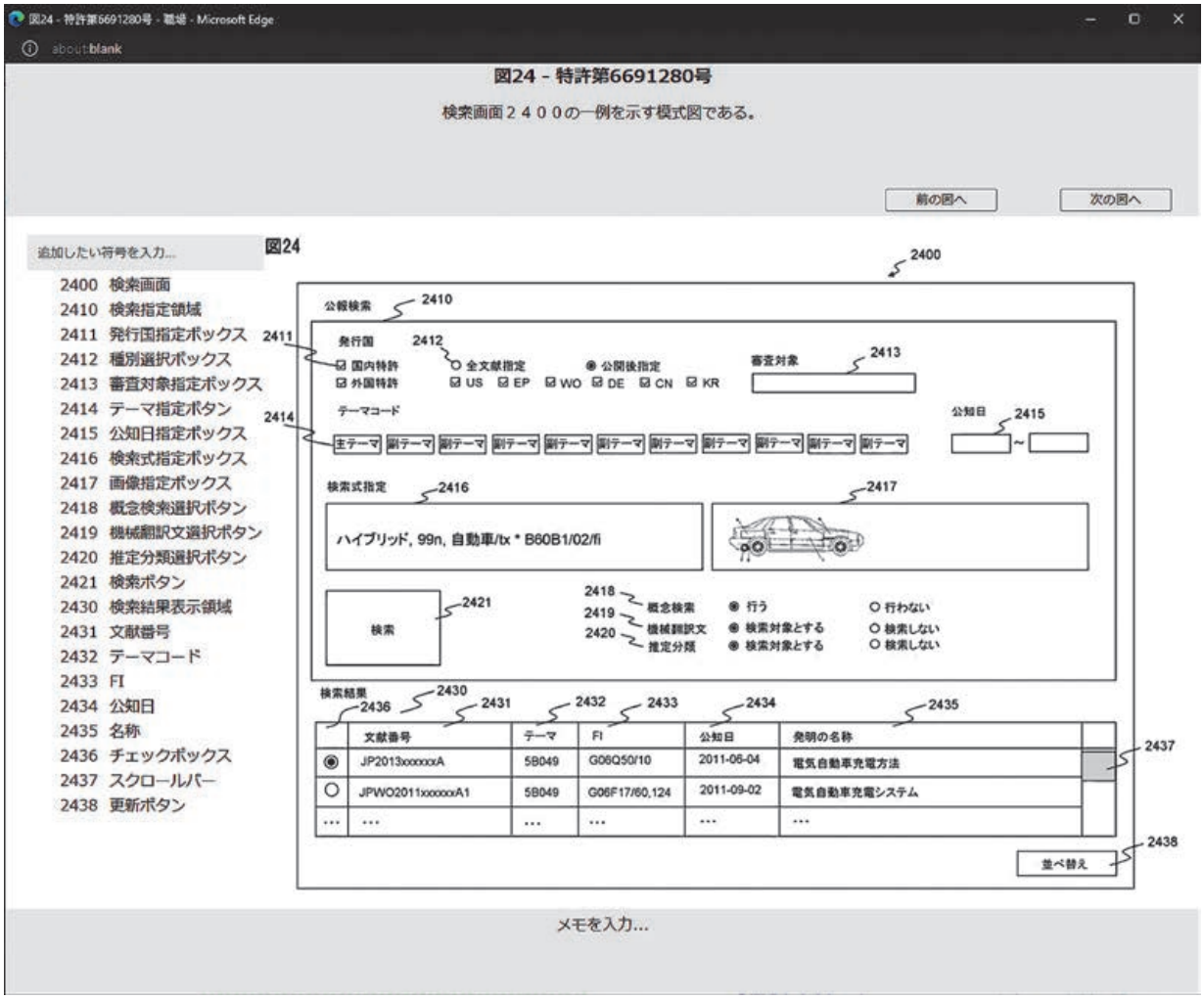

図3

図面をダブルクリックすると表示されるサブウィンドウの例。画面左側に符号と対応するテキストの一覧が表示され、画面右側に実際の図面が表示される。符号と対応するテキストの一覧は、画像処理及びテキスト分析により推定される。

---

7) 現時点で本ツールに直接実装しているのは太字にする機能と下線を付ける機能だけだが、Microsoft Wordなどで書式を変更してから本ツールにコピー＆ペーストすることで、フォントサイズなどの書式維持しつつ貼り付けることができる。ただし、少々面倒なので、どこまで手間をかけるべきかという問題はある。

3.3.5 図面のPowerPoint出力

それぞれの図面の下にはメモを入力するテキストボックスを配置しているが、図面が複雑な文献になると、やはり紙のようにフリーハンドで書き込みたいという場面が想定される。

このようなメモツールを自力でHTML上に実現するのは困難なので、代わりに、このツールからダウンロードできるHTMLファイルを読み込んで、各図面が一つのスライドに配置されたPowerPointファイルを出力するVBScriptのスクリプトを作成した。実際に出力されるPowerPointファイルの例を図4に示す。

PowerPointでは、フリーハンドで線を書き込む機能を始めとして様々な編集機能が実装されており、一般的なPDFリーダーよりも図面に対して柔軟な編集が可能である。タッチペンが使用できるデバイスを使えば、より利便性が増すかもしれない。（ただ、筆者はテキストによるメモで十分だと感じており、まだ本機能はあまり利用していない。）

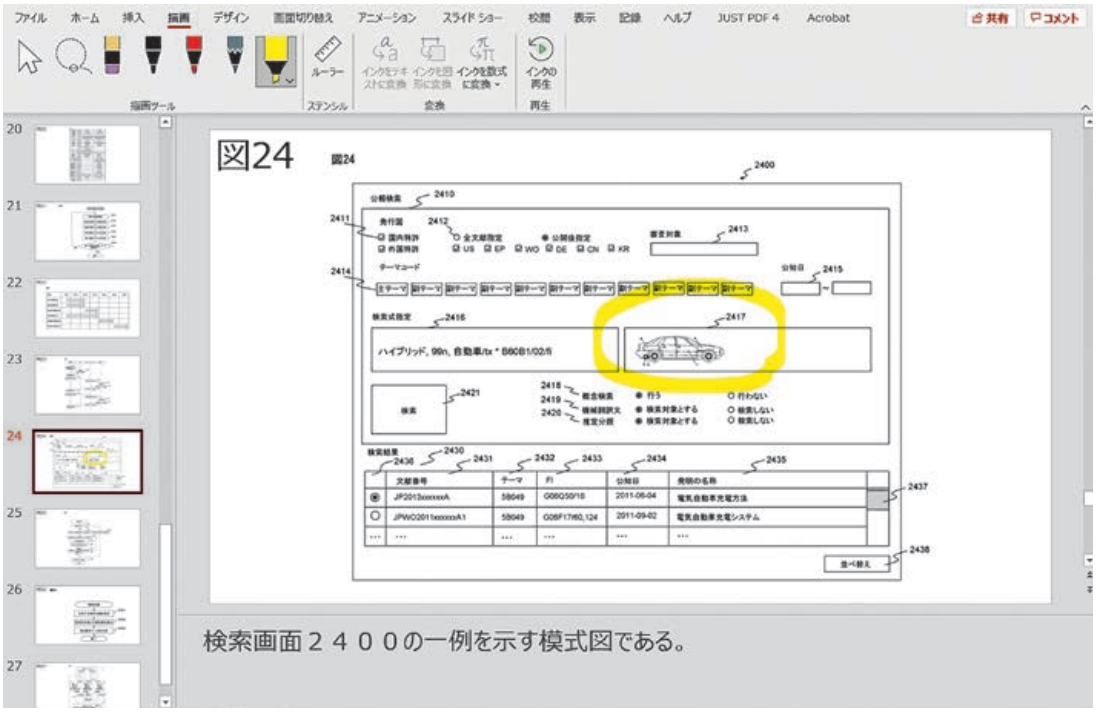

図4

VBScriptにより出力可能なPowerPointファイルの画面の例。フリーハンドでの書き込みを行いたい場合に有効か？

3.3.6 タイトル設定

画面左上のテキストボックスにタイトルを入力でき、入力したタイトルがWebブラウザ上のタブのタイトルとなる。複数の文献のHTMLファイルを開いている際に、どのタブがどのような文献を表すかをタイトルにより一目で把握できるようになり、大量の文献を扱う場合でも文献が散らからない（図5）。

また、HTMLファイルをダウンロードする際には設定したタイトルがファイル名となるように実装したため、フォルダ内で複数の特許文献のHTMLファイルを管理する際も整理しやすい。

適切なタイトルやファイル名を設定することで、大量のWebブラウザのタブやファイルの管理が容易になることは、紙審査に対するペーパーレス審査のメリットの一つであると感じている。

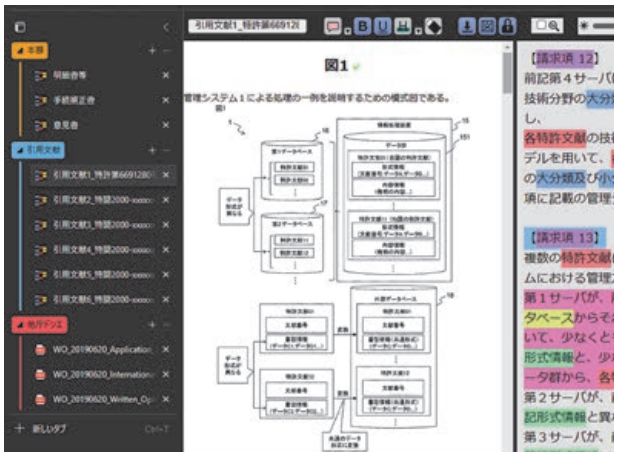

図5

タイトル設定の例。左上のテキストボックスに入力したタイトルがそのままタブのタイトルとなる。タブブラウザで複数タブを整理する際に有用である。（この図ではMicrosoft Edgeで垂直タブとタブグループを利用して整理している。）

3.4 ツール開発後の審査スタイル

図面が複雑でない分野を審査していることもあってか、筆者は本ツールの現状の機能で十分だと感じており、今では完全にペーパーレスで審査を行っている。特許文献に限らず、手続補正書や意見書などの審査経過書類も全て本ツールを利用して読んでいる。紙に印刷するよりも本ツール上で読む方が効率が良く、他のツール（補正の差分比較ツールなど）との連携が容易であり、紙より電子データの方が整理も楽であることから、登庁日でも基本的に印刷することはない。

ただ、ペーパーレスで審査するためには、ディスプレイを複数台使用できることが必要不可欠であると感じている。PCの画面の広さは、紙審査での机の広さに相当し、ワーキングメモリの大きさを表すものである。ワーキングメモリの大きさは処理効率に直結する。紙審査には、机の上に複数の書類を広げられるという大きなメリットがあり、ノートPC一台でこのメリットに打ち勝つのは難しい印象である[8]。

4. 特許文献以外への適用

4.1 英語論文

審査の先行技術調査において精読する文献は特許文献に限らない。筆者の担当する技術分野では非特許文献として英語論文を引用することが多いため、英語論文をペーパーレスで精読する方法も確立する必要がある。

英語論文は基本的にPDFファイルのため取り扱いが難しいが、本ツールを修正し、PDFファイルにスペクトルバーを付与するところまでは実装することができた（図6）。開発にあたっては、JavaScriptによりPDFを表示するためのライブラリであるPDF.jsを利用した[9]。書き込みを行う機能は実装できておらず、書き込みを行いたい場合は別途PDFリーダーを立ち上げる必要がある点が不便だが、スペクトルバーに慣れていることもあってか、英語論文の精読がかなり効率化された。

図6

英語論文のPDFファイルにスペクトルバーを付与した画面の例。

8) 仮想デスクトップなどのウィンドウ管理術を極めれば解決される問題なのかもしれない。

9) https://mozilla.github.io/pdf.js/

4.2 法律の条文

本誌の読者は、法律の条文の読解に苦労した経験が少なからずあるかと思う。本ツールの機能は特許文献の精読に有用であると感じたが、法律の条文に対して同等の効果を有するかどうか試すことにした。

図7は外国語書面出願について定めた特許法第三十六条の二の条文を示すものであり、図8は筆者が適当に設定したワードにより反転を行った様子を示すものである。ワードとしては、正規表現を使いつつ、項を示すもの（「〇条」、「〇項」）や期間を表すもの（「〇年」、「〇月」）、条文のキーとなりそうな用語（「外国語書面出願」、「翻訳文」等）を採用した。

ワード反転により、条文全体の構造を捉えやすくなるほか、他の条項の参照箇所も視認しやすくなり、全体として可読性が向上したように思う。また、この例は単一の条文を対象にワード反転を行ったが、特許法全体を対象とすることで、特定の用語（例えば「拒絶　理由　査定」など）に関連する条文を効率よく探すことも可能である。

現状、法律の条文の精読に資する機能としてはこの程度であるが、並列構造が複雑である長文を木構造で表示する、条文の引用関係や用語の定義などを分かりやすく表示するなど、ツールによる様々な支援の可能性があると考えている。

5. まとめ

ペーパーレス審査は、適切な環境の下では紙審査に優る部分が多いと筆者は考えており、特許文献の精読においては実際にペーパーレスの方が効率が良いという実感を得られるツールの開発に成功した。

審査においては、他の審査官との協議や決裁な

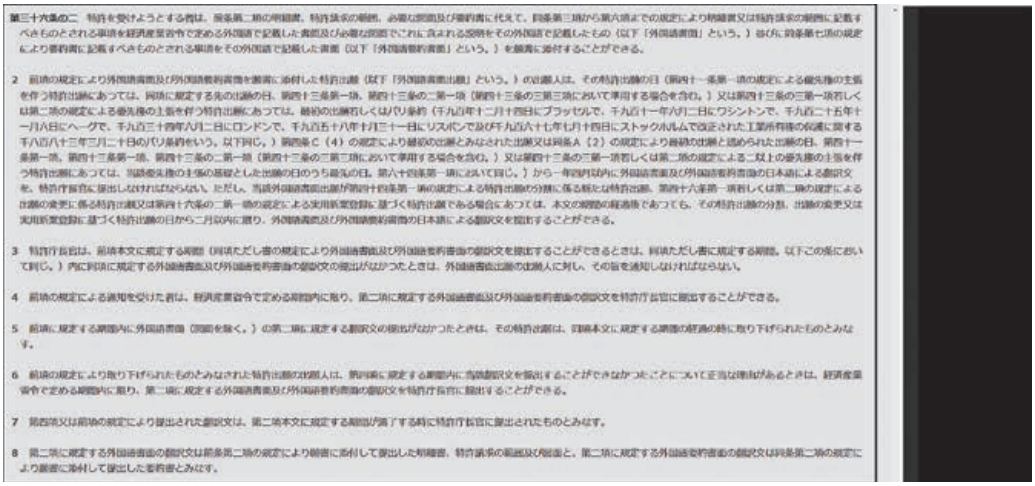

図7

ワード反転前の特許法第三十六条の二の条文。読む気が起きないのは筆者だけだろうか。

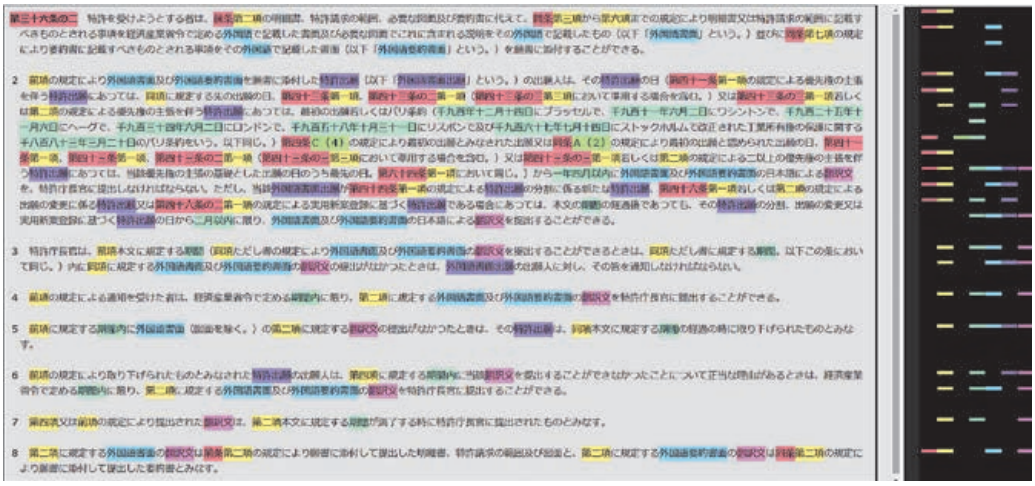

図7

ワード反転後の特許法第三十六条の二の条文。条文全体の構造が捉えやすくなっている。

ど、紙に印刷した方が負担が少ないと感じられる場面がまだ多く存在するが、審査官の創意工夫に加え、今後の適切なペーパーレス環境の構築により、ペーパーレスの方が便利だと感じる審査官や場面が増えていくものと期待される。

6. 番外編－自分用のツール開発について

筆者は入庁以来自分の業務に資するツール開発を続けている。この場をお借りして、自分のためのツールを開発することについて、筆者の考えるメリットやデメリットを述べさせていただく。

メリット１：業務への理解が深まる

ツール開発とは、自分がコンピュータにさせたい処理をコードの形で表現することである。そのためには、業務課題を正確に把握したうえで、課題の解決に必要な作業を特定し、作業手順を客観的に書き下すことが求められる。この作業を行う過程で、自分の業務への理解が深まる。場合によっては、そもそもツール開発の必要がなく、作業フローを変更すべきとの結論に到ることもある。

ツール開発の経験がない場合でも、異動にあたって引き継ぎ資料を作成する際に似たような作業が求められるように思う。課題を整理し、それらの課題に対して取り組んでいる業務をまとめ、(特に係長レベルの引き継ぎ資料では) 何らかの具体的な作業手順をまとめる。このプロセスを経て、自分の業務を大きな視点から見直すことで、異動直前に改善点が見つかったり、そもそもゴールが明確でないことに気付いたりする経験がある方も多いのではないだろうか。

よいツールを開発できるということは、業務を深く理解できていることを意味すると筆者は考えている。プログラミングスキルも当然必要であり、また優れたスキルを身につければそれだけ解決手段の引き出しが増えるが、それだけあれば足りる訳ではない。この点がツール開発において難しく、また面白いところである。

メリット２：フィードバックループが高速に回る

自分用に開発したツールは、開発者と使用者のいずれも自分であり、開発者と使用者との間のコミュニケーションコストが発生しない。そのため、使用者としての自分が気付いた不具合や改善のアイディアをすぐにツール開発に活かすことができる。日常的に使用するツールであれば、普段の業務がそのままツールのデバッグや評価を兼ねることになる。(いわゆるドッグフーディングと似た状態かと思う。)

自分が欲しいツールなのだから、欲しいと思った機能を実装して終わりのようにも思えるが、一度作ったツールを使うと不満点や改善点が多く見つかり、コード修正と試用のループが続くことが多い。

使うのが自分である以上、開発者としてもより良いツールを開発する高いモチベーションが保たれる。努力した分だけ業務の効率化やスキル向上という形で報酬が得られ、それにより新たなツール開発の可能性が広がっていく。

デメリット：信頼性や保守性が低い

ここまで自分用のツール開発のメリットを挙げてきたが、組織として開発したツールより信頼性や保守性の点で劣る点は無視できない。比較するのもおこがましいかもしれないが、個人で作成したツールはオープンソースソフトウェアと似ている部分があり、組織がオープンソースソフトウェアと付き合うにあたり生じる諸問題と同様の問題をはらんでいると考えている。この問題への良い対処法は思い浮かばないものの、個人でツールを作成する際には少なくとも信頼性や保守性のデメリットを認識しておくことが重要かと思う。

以上の特性をふまえたうえで、今後も目の前の業務課題に応じたツール開発を続けていきたいと考えている。

profile

樋口 祐介 (ひぐち ゆうすけ)

2015年４月　特許庁入庁 (審査第一部応用光学)
2018年４月　審査官昇任
2019年７月　調整課審査企画室 審査企画第一係長
2021年１月　審査第一部分析診断 審査官
2021年12月　調整課審査企画室 (４ヶ月間)
(2022年７月～　コート・ダジュール大学 客員研究員)