VersatileUX

A Disruptive Multimedia Application-Plugin Framework for Test Preparation and e-Learning

Linguistic Technology Systems (LTS) has designed a novel Application-Plugin Framework to improve test preparation and e-Learning by enhancing document viewers with multimedia features, such as **3D** graphics, audio, and video. This plugin framework would allow document viewers (e.g. **PDF** viewers and e-Book readers) to launch — and share data with — a diverse array of applications that exist for both scientific and social scientific (and humanities) disciplines.

By using this plugin framework, document viewers would be able to support interactive and multimedia reading/studying experiences to an unprecedented degree. Students preparing for exams would have at their disposal stimulating multimedia presentations that offer sophisticated data visualization and **3D** graphics tools, customized for individual subjects: e.g. **3D** molecular models for chemistry; or **3D** tissue models for biology.

Moreover, in addition to offering multimedia features, such plugins could include instructional features — review questions, assignment instructions, or definitions of important concepts — neatly appearing in dialog boxes while the student is engaged in reading the course material.

The versatility of this plugin framework signifies a major boost to **UX** functionality. For instance, metadata logging the student's usage of e-learning programs can be easily shared between the application (such as a chemistry application that shows **3D** molecular structures) and the document viewer. As a result, such test preparation and e-Learning plugins would be able to keep close track of the student's e-learning process and serve as a guide to the student.

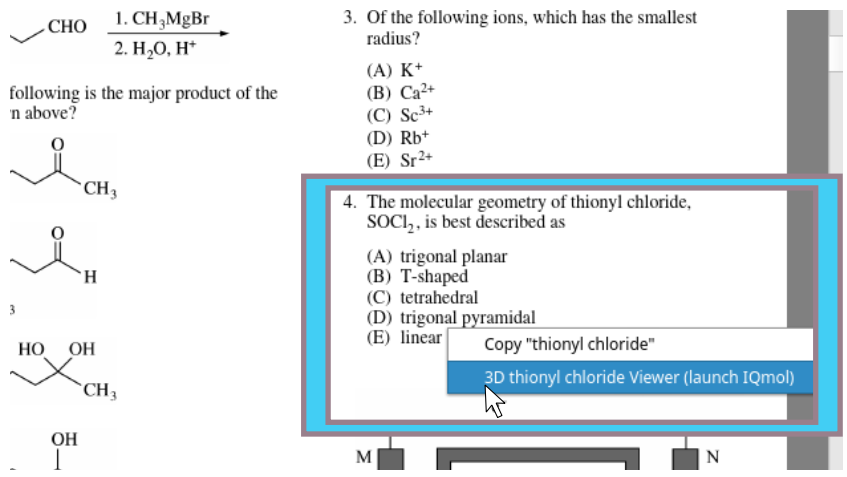
How this Framework would Work in a Science Application

Our Educational Plugin framework refers to a toolkit for implementing multiple plugins to be embedded in many different scientific and social-scientific applications. One feature unique to this framework is that individual/distinct Educational Plugins would be able to communicate with one another. For example, plugins for document viewers should be able to send data to plugins for scientific applications. In this way, students, with a single click, would have access to multimedia content linked to the e-Learning materials that they are currently studying, without the cumbersome steps of having to leave one application to open another.

Illustration:

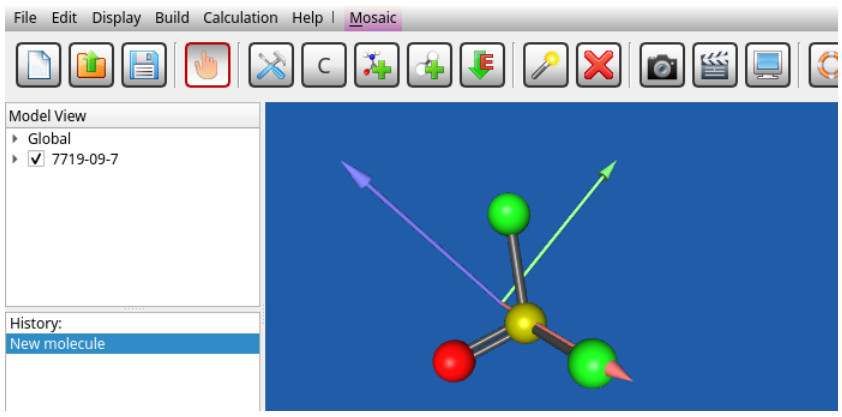
For a concrete example of advanced functionality that can be achieved by connecting two distinct plugins, consider a student reading through a **GRE** practice test in Chemistry, such as that published by the Educational Testing Service. This book has sample multiple-choice questions such as (on page 11, number 4), "**The molecular geometry of thionyl chloride, $SOCl_2$, is best described as** *(A) trigonal planar, (B) T-shaped, (C) tetrahedral, (D) trigonal pyramidal, or (E) linear*". To understand this question and its corresponding multiple-choice answers, it would help students if they were able to view a **3D**

Figure 1: A Thionyl Chloride Question in XpdfReader



model of thionyl chloride, which can be done with the aid of molecular visualization software, such as IQMOL. To support this functionality, our plugin embedded within a document-viewer application (here XPDF) would launch IQMOL, sending data through a corresponding Educational Plugin embedded in IQMOL. Specifically, question *4* in the GRE practice test may be associated with a Molecular Data file for $SOCl_2$; the XPDF plugin would launch IQMOL, sending along a data package identifying this $SOCl_2$ file to IQMOL's own plugin, with instructions relayed to the program to load the file into an IQMOL session. The student could initiate this process by selecting a context-menu option (called "3D thionyl chloride Viewer" in Figure 1; please note the highlighted option to which the cursor is pointing). The end result, then, would be that the student, with a single click, has access to an interactive 3D graphic representing thionyl chloride. The same functionality would be available for any chemical compound which has associated data in formats such as Chemical Markup Language or Protein Data Bank (*see* Figure 2).

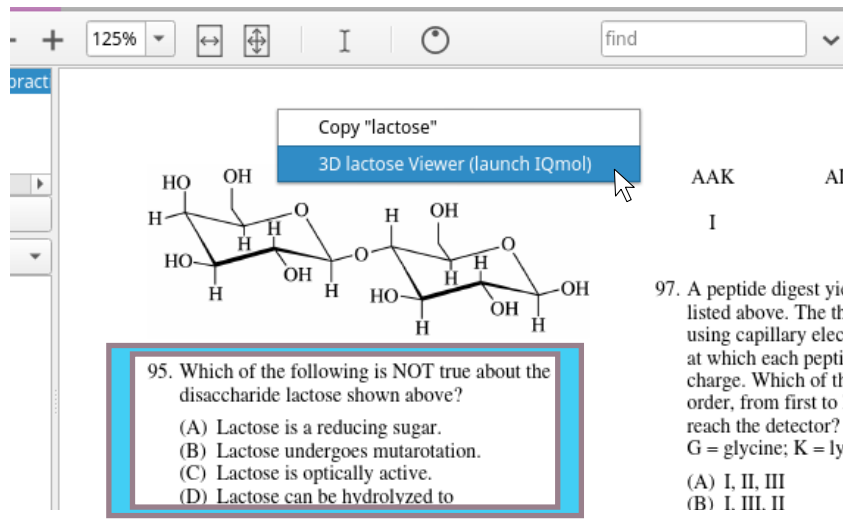
Figure 2: Thionyl Chloride in IQmol (drawn from Protein Data Bank)



Keeping Track of the Student's Actions Performed on the Application – a Novel Feature to Enhance e-Learning:

The previous illustration involving Thionyl Chloride exemplified a simple data structure transmitted between Educational Plugins: specifically, the name of a single file to open. In some cases, however, the information sent between plugins might be more complex and detailed. To accommodate this, all Educational Plugins in our framework would be endowed with a *common* vocabulary for representing multi-part data structures. For example, if a student views a *second* molecule in IQMOL, the document viewer should identify not only *that* file, but any *previous* files they had viewed, wherein the student could conveniently refer back to previous files, as needed. This would be the case where a student reading through the GRE Chemistry practice exam chooses to launch IQMOL a second time — in conjunction with a later question such as number *95* in the test (*see* Figure 3 above), which is about the molecular structure of lactose.

Figure 3: A Lactose Question in XpdfReader

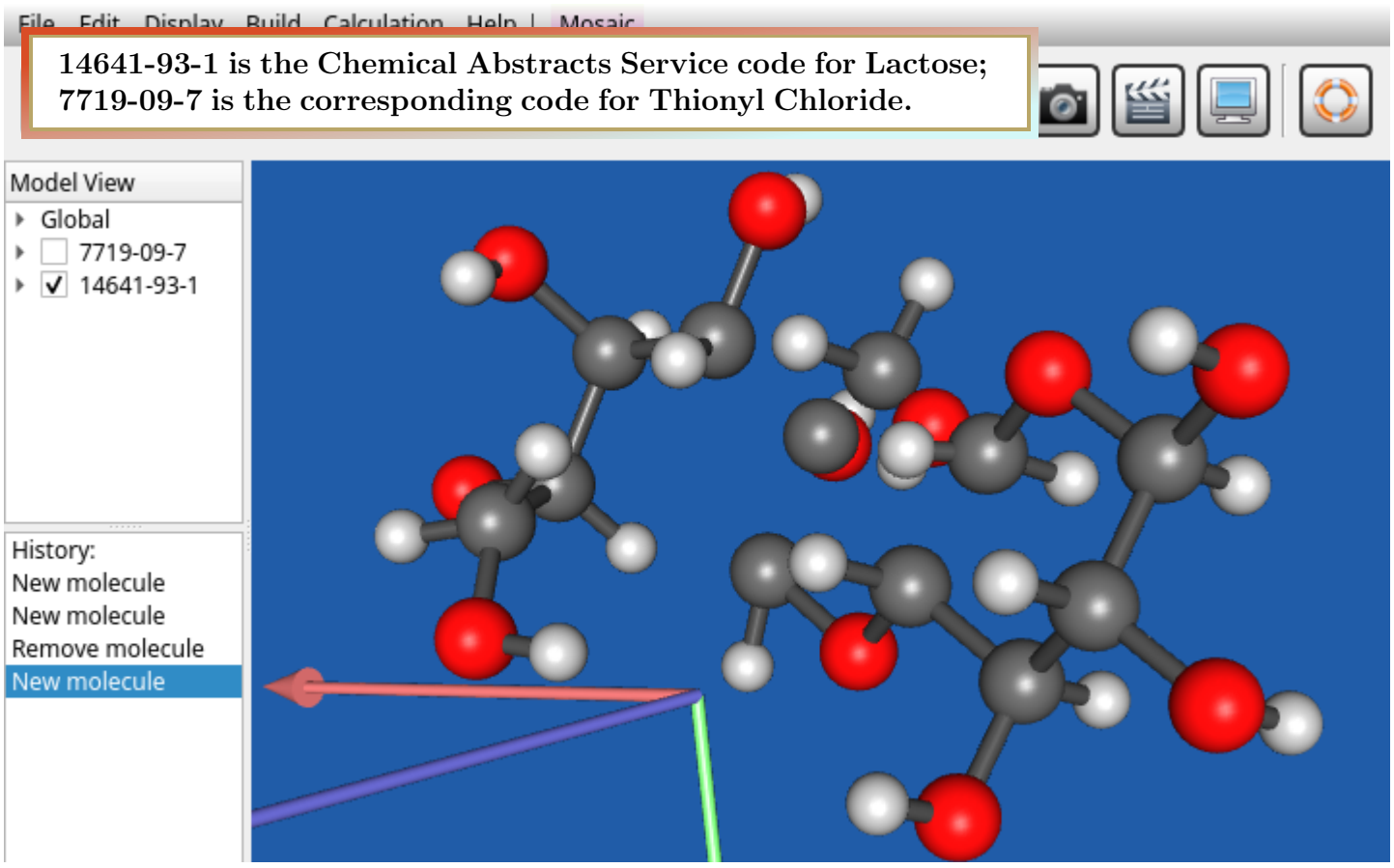


In this case, if the student wishes to refresh their recollection of the molecular structure(s) seen in a prior session, the plugin would be able to send information not only about the present (lactose) request but also about the $SOCl_2$ (Thionyl Chloride) file that the student had viewed earlier, thereby *keeping track of their actions* performed on the application. The student's viewing history is visible within the Model View panel at the top-left on Figure 4 (below), where the Thionyl Chloride file is listed (via the corresponding number for this inorganic compound as designated by the IQMOL application) above the *checked* lactose file box, which represents the molecular compound currently being seen by the student in view-port.

This offers a valuable UX feature to our Application Plug-in Framework because the plugin would keep track of the student's usage-history — sent in the form of metadata over to the science application — so that the student would immediately be able to view the molecular structures they had previously viewed in IQMOL. This would be done by simply clicking on the appropriate box for each molecular

structure listed on the Model View panel. In short, by being able to return to the 3D graphic of the Thionly Chloride, the molecular structure that had been studied earlier, the Application Plugin Framework boosts UX by enabling the student to refresh their memory of the graphic rendition of the molecular structure they had seen in the prior session when presently examining the molecular structure of the new compound.[1]

Figure 4: Lactose in IQmol



The Tools that Comprise our Application Plugin Framework

The essential goal of e-Learning is to equip students with digital content which they can use to learn new material or review concepts they have already encountered. In fact, e-Learning is an important and expanding technology market because interactive digital teaching materials are conducive to an engaging and exploratory kind of learning, which many have found outperforms conventional teaching materials (such as textbook chapters) that engender a more static and passive learning style.

All things considered, actually achieving an interactive learning environment — at least when implemented via digital tools such as PDF viewers and scientific applications — requires special-purpose programming across several different software components. The tools that comprise our framework are designed specifically to enable functionality to be introduced within document viewers (such as PDF viewers) and scientific applications (such as IQmol) so that students have various ways of exploring and interacting with e-Learning materials. Once they have chosen to pursue a particular kind of exploration — for example, if they decide to view a 3D graphic displaying the molecular structure of a chemical compound which they need in order to understand how its chemical formula is configured — our framework ensures that their e-Learning software provides all requisite features allowing students to accomplish the tasks at hand in the most user-friendly manner. In this way, the e-Learning and test preparation would be made as easy and seamless as possible so as not to add to the student's existing burden of studying new course material and preparing for exams.

For example, in questions *4* and *95* (as illustrated above) the relevant action would be an option to view the molecular files in the IQmol science application, showing the unique molecular structure of each compound. Plugin data would be needed in order to map the textual boundaries of the question (and its menu list of multiple-choice answers) to on-screen coordinates, so that context menus can be customized for each question by the teacher/instructor who prepares the course materials. The plugin

[1] The usage-history features described here are specific to IQmol; other applications have different models for usage-history and application/session state, but in every case a student's prior actions can be conveyed via Educational Plugins (as metadata) so that applications can easily reload sessions upon being launched.

must access this information in order to automate the process of downloading files in the Protein Data Bank (PDB) format from the Chemical Abstracts Service (where molecular databases are warehoused). The role of the Educational Plugin in this case — referring to the example of the student's choosing to view 3D graphical depictions of molecular structures — would be to identify the specific molecule which the student is studying (based on the current on-screen context-menu cursor location) and use that information to download PDB files and then display them via IQmol.

Comparing our Educational Plugin Framework with Conventional Methods:

In the case of IQmol, students without the aid of a plugin would first need to obtain the code number for each compound, which they would find on websites such as Chemical Abstracts Service, so as to begin a complicated process of finding the visual image to show the molecular structure of the compound they are studying. In addition, this foray through a maze of complicated steps would require the student to leave the article/chapter material they are presently studying so that they can then go over to the Chemical Abstracts site to look up the physical structure of the compound. Now that's just for starters, because once on the new website, the student would have to input the name of the compound and retrieve its corresponding code number so that they could begin downloading the files that correspond to the molecular structures to save the files to the their disk. Then, they would need to launch IQmol and finally open the saved files manually.

However, by eliminating each of these time-consuming manual steps in favor of just one single context-menu action — triggering a behind-the-scenes orchestration of automated steps culminating in IQmol being launched with the precise molecular graphic rendering on display right then and there — the Educational Plugin would make the e-Learning experience streamlined and easy: with one single action (such as one context-menu selection) the plugin automatically takes care of a complex sequence of tasks, and spares the user from having to leave the application where they are engaged in the study of course material. In essence, such versatile UX allows the student maximum concentration on the course material at hand, by removing the distractions that inevitably occur when having to navigate such features manually.

Then, when a document is opened, the Educational Plugin would extract the embedded file so as to read plugin-specific data about the document — in particular, to identify PDF coordinates for document elements requiring special plugin actions (such as launching IQmol). In short, our Educational Plugin would do all this work for the student behind-the-scenes so that the student can have a pleasant (and non-distracting) interactive experience with the document viewer, wherein the plugin framework is embedded.

How Our Educational Plugins Embed Metadata in PDF Files and Enhance UX:

In order to automate software tasks thereby relieving the student of the burden of taking so many extra steps in the e-Learning process, our Educational Plugins are specifically designed to embed metadata structures within PDF files (or files in other formats, such as ePub or HTML) by annotating such files with instructions that create an association between the e-Learning document and a scientific application (such as IQmol). This metadata can then be leveraged so as to implement e-Learning features which convert scientific applications into excellent pedagogical tools. For example, IQmol's built-in algorithms for calculating molecular forces can inspire digital lessons explaining the formulae for molecular bond formation. In so doing, any portion of a chemistry-related e-Learning resource which deals with molecular bonds could, therefore, be annotated with instructions that would cause IQmol's features involving computations and visualizations of molecular bonds to be highlighted. As this example illustrates, our Educational Plugins are deliberately designed with such agility/versatility so as to optimize the features of each scientific application in order to add crucial instructional annotations

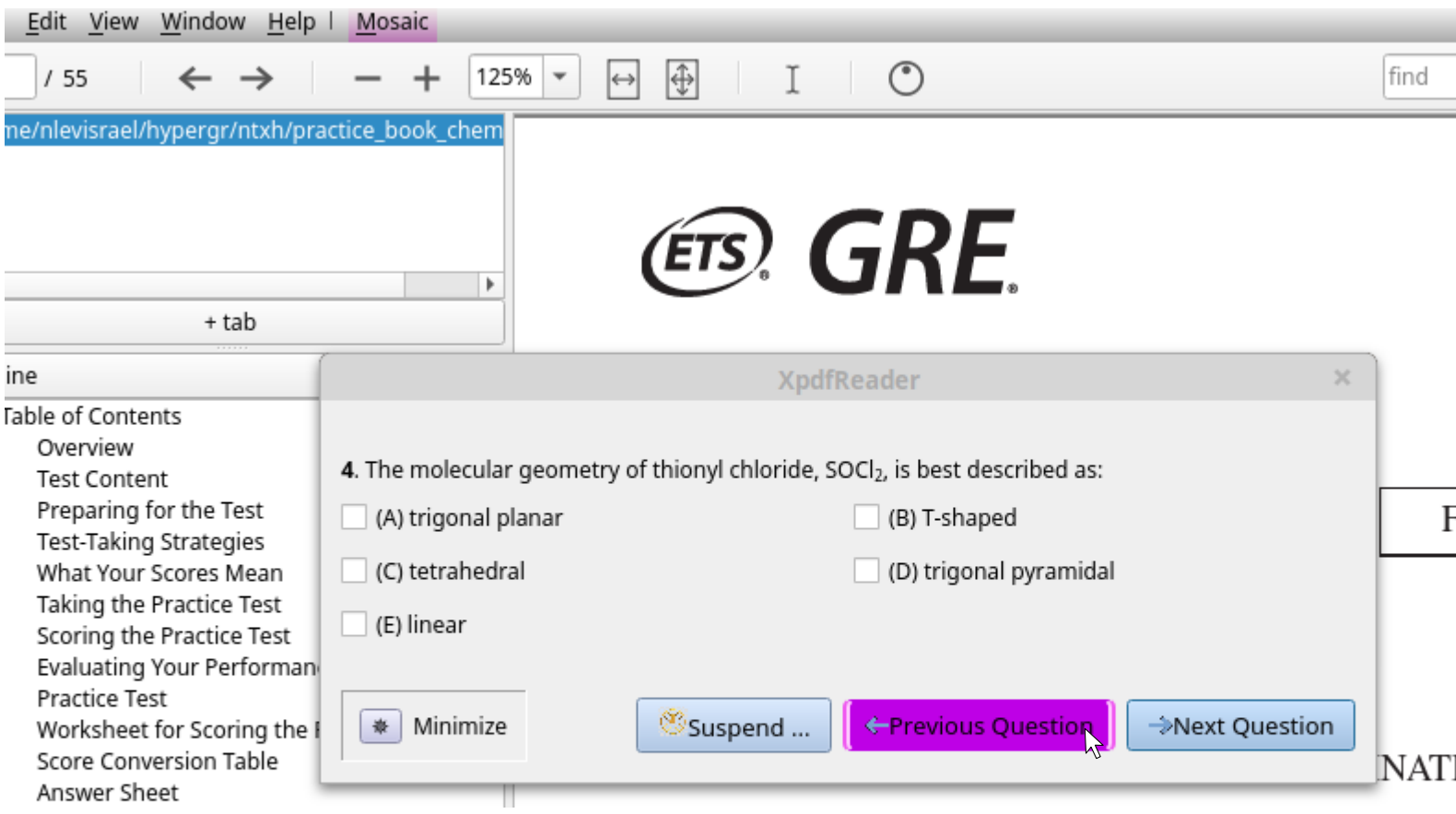
that enhance the learning experience for the student and make their test preparation experience more user-friendly.

To support the automated execution of software tasks — such as downloading files and launching an application that is able to open such files — our educational plugins contain specific tools that help compose publications that embed document metadata. We use our own coinage, "Semantic Document Infoset" (**SDI**), to refer to the structured compilation of document metadata. As such, our **SDI** would effectively divide written text into textual units (e.g. subsections, paragraphs, sentences, quotations, and bullet lists) and also identify document elements, such as technical terms — which may be compiled into a glossary — and figure illustrations.

Using this format, the plugin code can then examine a publication's **SDI** to generate machine-readable structural representations of publication text, which document viewers may use to augment the underlying document with additional instructional and/or multimedia features — review questions, student instructions, glossaries, reading assignments, **3D** graphics, and so forth. In addition, the **SDI** may be used to guide plugins when sharing data between applications — in Figure 1, for instance, selecting the Molecular Data file in the **PDF** viewer to send to the **IQmol** chemistry application based on the screen coordinates of the context menu — but also to enhance the presentation of content within the host application, such as **IQmol**. For example, Figure 5, below, shows how a plugin could provide an alternative interface to the student for viewing practice-test questions, where students can consider one question at a time, isolated in its own window, which may help them to singularly focus their attention on each question in turn. Our plugin framework would accomplish this by instructing the application to place each question in its own individual dialog box so that the student only sees one question at a time and is, therefore, not distracted by the prior or subsequent questions.[2]

In short, our Educational Plugin Framework is designed to maximize the effectiveness of e-Learning and test preparation by providing the framework that enables a wide range of pedagogical customization. In so doing, the intuitive skills of instructors/teachers — providing an innate sense of how to best organize study material so that the student can focus properly — can be exploited for optimal design of e-Learning and test preparation materials that produce a rewarding user-experience and better academic performance.

Figure 5: Each Question Appears in XPDF in its own Dialog Box without the Distraction of Prior or Subsequent Questions



[2]Educational Plugin implementations can include LaTeX packages which automate the creation of SDI data (placed as an embedded file in the generated PDF document). This embedded data can then be read by plugins to compose multi-application networking requests, populate question/answer windows, or introduce other kinds of teaching content: review questions, glossaries, discussions of figure illustrations, etc. In documents where questions are printed as part of the publication text (for example, GRE practice tests), the LaTeX code can store the PDF coordinates for the questions so that the document automatically scrolls while students work their way through a practice test session. These techniques may also be used to add review questions/answers to those publications which are not expressly designed as test-prep materials (such as textbooks and research papers), with the question/answer windows that are currently in view being synced to relevant sentences or paragraphs in the publications.