

CSE 6390E Computational Linguistics Project  
Winter 2010  
"CL Book"

Dmitri Shuryalov  
shuryork@cse.yorku.ca

Ameeta Agrawal  
ameeta@cse.yorku.ca

## Table of Contents

1. Introduction .....	3
1.1 Motivation .....	3
1.2 Challenges .....	3
1.3 Outline of the report .....	3
2. System.....	4
2.1 English Grammar .....	4
2.2 Translating from English to other languages .....	4
2.3 Implementation .....	5
3. Examples of Output.....	7
4. Conclusion and Future Work .....	16
5. System Manual.....	17
5.1 Deployment.....	17
5.2 Functionality .....	18
6. References.....	21
7. Appendix .....	21
7.1. The Penn Treebank POS tagset .....	21
7.2 CL Lessons.txt .....	22

# **1. Introduction**

Ever tried learning a foreign language? Have you noticed how the books you *could* read were often boring? And were the books you *wanted* to read just that bit too hard to understand? Did you wish for a quick translation of a complex passage? Or wished there was a quick way to learn English for beginners?

With paper books, you're pretty much stuck. But e-books—with the right combination of software and open formats—can help keep you reading and learning in the new language.

In this project we designed a system that helps as English grammar learning software for beginners as well as translation software for ebooks.

## **1.1 Motivation**

Grammar learning through real examples – Most of the texts provided for learning grammar are boring and feel artificial. How about being able to choose your own text and have the reader software automatically highlight the structures you are learning this week, whether it is proper nouns, present perfect constructions or conjugations of the irregular verb 'to be'? With the material being presented completely in context, the rules will be easier to understand and recall. And even if you are rereading the last week's passage, you are learning something new, as the highlighted parts will change.

Parallel texts – Intermediate and advanced readers appreciate being able to read original text, while still having a good translation available a glance away.

Paper books like this do exist, but just a few, due to a high cost of production and distributed target market. For e-books, the ever decreasing price of the storage makes the size of the download irrelevant – slashing the cost of physical production. And with electronic distribution, the market reach is as wide as the internet itself.

## **1.2 Challenges**

In this project, we use free available Stanford POS tagger library and Google Translate library for some processing. This means the accuracy of the results of our program is dependent on the accuracy of the libraries we use. However, with Stanford POS tagger's 96.92% accuracy and Google translate's pretty high accuracy, our program gives good results overall.

## **1.3 Outline of the report**

The report starts by describing the system and the implementation details in section 2. It covers the data structures and the algorithms used. It also describes in detail the Grammar learning and the Translation parts of the program. Section 3 includes sample output of the program with brief descriptions. Section 4 contains the concluding remarks and ideas about future work. Section 5 outlines the System Manual and section 6 annotates the references. Lastly, section 7 contains an appendix.

## 2. System

Our system CL Book has two major components: English Grammar Learning and Translating. We examine each component below.

### 2.1 English Grammar

The program provides several English grammar topics for beginners. These have been adapted from a level 1 English grammar topics [3][4]. The lessons include Adjectives, Adverbs, Determiners, Nouns, Pronouns, Passive Voice, Possessive, Conjunction and Verbs.

This part of the program can be broken down into steps such as:

- Preprocessing text
  - [Removing Newline Characters from Paragraphs](#) as explained in section Algorithms.
- Using MaxentTagger from Stanford POS Tagger API
  - Stanford POS Tagger: A Part-Of-Speech Tagger (POS Tagger) is a piece of software that reads text in some language and assigns parts of speech to each word (and other token), such as noun, verb, adjective, etc., although generally computational applications use more fine-grained POS tags like 'noun-plural'. Stanford POS Tagger software is a Java implementation of the log-linear part-of-speech taggers described in [1] and [2]. The tag notation is taken from the Penn Tree Bank POS tagset described in [5], table 2 ,page 317. Refer [Appendix POS tagset](#).
  - We use the already trained *left3words-wsj-0-18.tagger* model for tagging. This model uses only left sequence information but less unknown words and lexical features. Its accuracy was 96.92% on Penn Treebank WSJ secs [1][2].
  - The input to the tagger is a simple String e.g. *I wonder if I shall fall right through the earth!*
  - The output tagged text can be produced in several styles. The tags can be separated from the words by a character, which you can specify (this is the default, with an underscore as the separator), or you can get two tab-separated columns (good for spreadsheets or the Unix `cut` command), or you can get ouptput in XML. We use the format `<word_POStag>` e.g. *I\_PRP wonder\_VBP if\_IN I\_PRP shall\_MD fall\_VB right\_JJ through\_IN the\_DT earth\_NN !\_.*
- Processing the tagged output for each lesson topic
  - We separate the word and the tag by tokenizing the String using the underscore delimiter.
  - Then we find the start and last index of the word in the entire String. This will be used later when highlighting the text in the reader.
  - The indices and the tag are then stored in a Map.

### 2.2 Translating from English to other languages

The program provides translation facility by tapping into the Google Translate API. The text can be translated from English to any language that is supported by Google Translate e.g. Arabic, Chinese, French, German, Hindi, Italian, Persian, Polish, Russian, Belarusian, Serbian, Thai and Swahili to name a few.

## 2.3 Implementation

### 2.3.1 System Requirements

- Java 1.6+
- 120+ MB of RAM
- Internet connection to the Google Translate server for translating

### 2.3.2 Data Structures

We have used a small number of Data Structures in our implementation of CL Book. They are used for tasks, such as:

1. Keeping the Loaded Book in Memory
  - When a book is loaded, we need to break it up into pages. The source ebook file is a plain text file, with newline characters at the end of each line. We take 24 lines of text and combine them into one book page. We store each book page as a String, and a Vector of Strings contains all the book pages.
2. Word Tagging Data for Current Page
  - The output from the tagger is a String of words and their tags. This is tokenized to retrieve the first and last index of each word as it appears in the original text. This first and last index pair is called a Tuple. Finally, a Map of type <Tuple,String> is used to save the results. This makes it easier when we need to highlight the words in the display. Furthermore, two arrays are used: one of type Tuple and the other of type String. These save the Tuples and tags respectively. This is needed when we check the tags of the previous or next few words for computing dependency between words as in the lesson "Passive Voice".

### 2.3.3 Algorithms

There are various algorithms used throughout CL Book project.

1. Loading Book to Memory
  - To load a book, which is given in a plain text format, we read the file one line at a time. Every 24 lines are concatenated into a page, and the page is added to a Vector of Strings containing all the pages.
2. Removing Newline Characters from Paragraphs
  - After a book is loaded, each page consists of 24 lines of text. There are newline characters within paragraphs. For multiple reasons, we must remove those newline characters before further analysis of the text. To do so, an algorithm is employed. It looks through each line of text, and adds it to the current block. When it sees that the next line is blank, it knows the current paragraph is over, so it finishes the current block and starts a new one.
3. Processing Text with POS tagger
  - For each entry in the [Map](#), we check the value i.e. the POS tag. Depending on what lesson the user has selected, we create a list of Tuples (start and end indices of words to highlight) that correspond to the selected lesson. For e.g. if the user selects the lesson "proper nouns", then all proper nouns will be highlighted in the text currently displayed

on the screen. For general part of speech such as nouns, verbs, pronouns, the tagger's tags are pretty straightforward. For more complex lessons such as comparative adjectives, comparative adverbs, passive voice, possession, we need to take into account the neighbouring words' dependencies.

- In a phrase such as "more beautiful", the tagger tags "more" as *JJR* (syntax for comparative adjective) and "beautiful" as *JJ* (syntax for adjective). So when we would like to display all the examples containing "comparative adjectives", we need to highlight not only "more" which the tagger tagged correctly, but also "beautiful". In order to do that, we need to check whether the next tag after *JJR* is *JJ*. Similarly we can compute comparative adverbs.
- For passive voice, we first look for a tag *VCN* which stands for past participle form of a verb. Then we check if the preceding word is either *VBG* (gerund form) or *VBZ* (third person singular present) or *VBD* (past tense) of verb "TO". If yes, then the phrase can be classified as passive voice structure.

#### 4. Adding Newline Characters to Balloon Tooltips

- When displaying Balloon Tooltips, HTML is used to specify the font face, size, as well as the newlines. Without line breaks, the entire tooltip ends up being too long and doesn't fit on the screen. However, the source text for a tooltip comes with no newline characters (because they've already been removed earlier). So an algorithm is used to recreate these newline breaks, by adding the `<br>` HTML tag at appropriate locations. It scans through the source text one character at a time, and when it counts 80 characters within the same paragraph, it adds a `<br>` tag. It repeats this process until the entire source text is converted.

#### 5. Translate with Preserved Formatting

- Unfortunately, our translator service drops any formatting in the source text when returning the translated result. Thus, an algorithm that counteracts this had to be created. Otherwise, all paragraphs merged into one, without any line breaks in between, which was unacceptable. In order to solve this, an algorithm similar to the "Removing Newline Characters from Paragraphs", but with some intermediate steps added. It begins by breaking up the text into blocks. Next, it translates each block separately, one by one. Afterwards, it reassembles all the translated paragraphs into one final combined result.

### 2.3.4 Resources/Files

#### Lessons Description File

We have used a custom file format to describe all of our grammar lessons that you find available within CL Book. The file is named "CL Lessons.txt" and resides in the root folder of the CL Book project.

The format of this file is as follows. For each lesson, the first line of text is the lesson name (e.g. "Lesson 1: Adjectives"). The second line of text is the internal id of the lesson (e.g. "adjective"). The following lines of text, up until a blank line, is the HTML description of the lesson.

This is a better solution than hard-coding all that information due to the extra flexibility it affords. It has allowed us to continue adding lessons, and modifying their description without having to edit code, working outside of the Java project. Refer [Appendix CL Lessons.txt](#).

### 2.3.5 Alternative implementation ideas

We preferred using Java language since the two libraries (Stanford POS tagger and Google translate) that we're using have Java APIs. But an alternative would be to use C++ and OpenGL for better visual rendering. Moreover, currently the program may use upto 100 MB of memory. Coding in C++ *might* make it use less memory.

As for the algorithms, we would have liked to design a better idea for processing tagged String. Currently, we check each tag and its neighbouring words' tags individually by looping over them and using AND and OR conditions. However, we would like to improve it by using an idea where you could search for a tag in some given range of a word. For e.g. if a word is tagged as a "past participle", you'd like to know the tags of the previous and the next three to five words. A possible data structure could be a multi-map.

## 3. Examples of Output

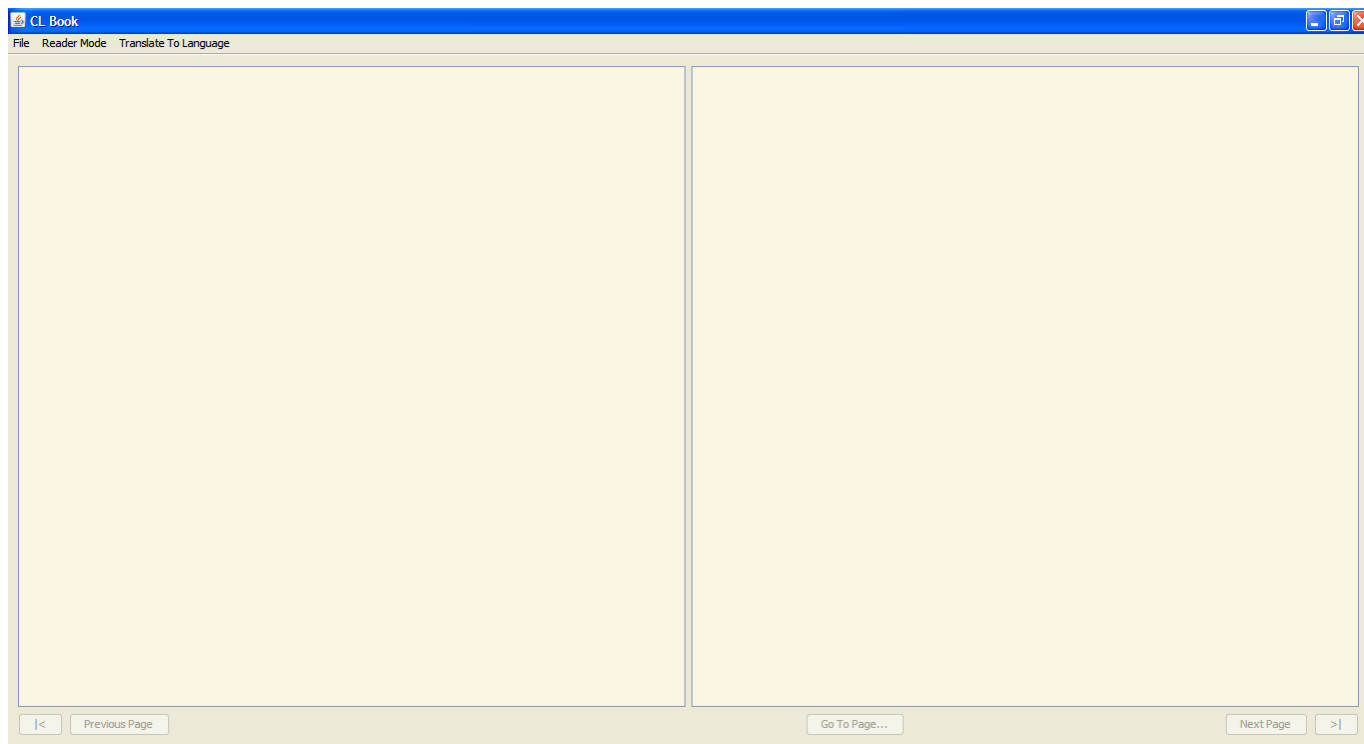


Figure 1. Blank Canvas

This is what the CL Book reader program looks like upon launching. You are presented with a big window, but no book is opened yet.

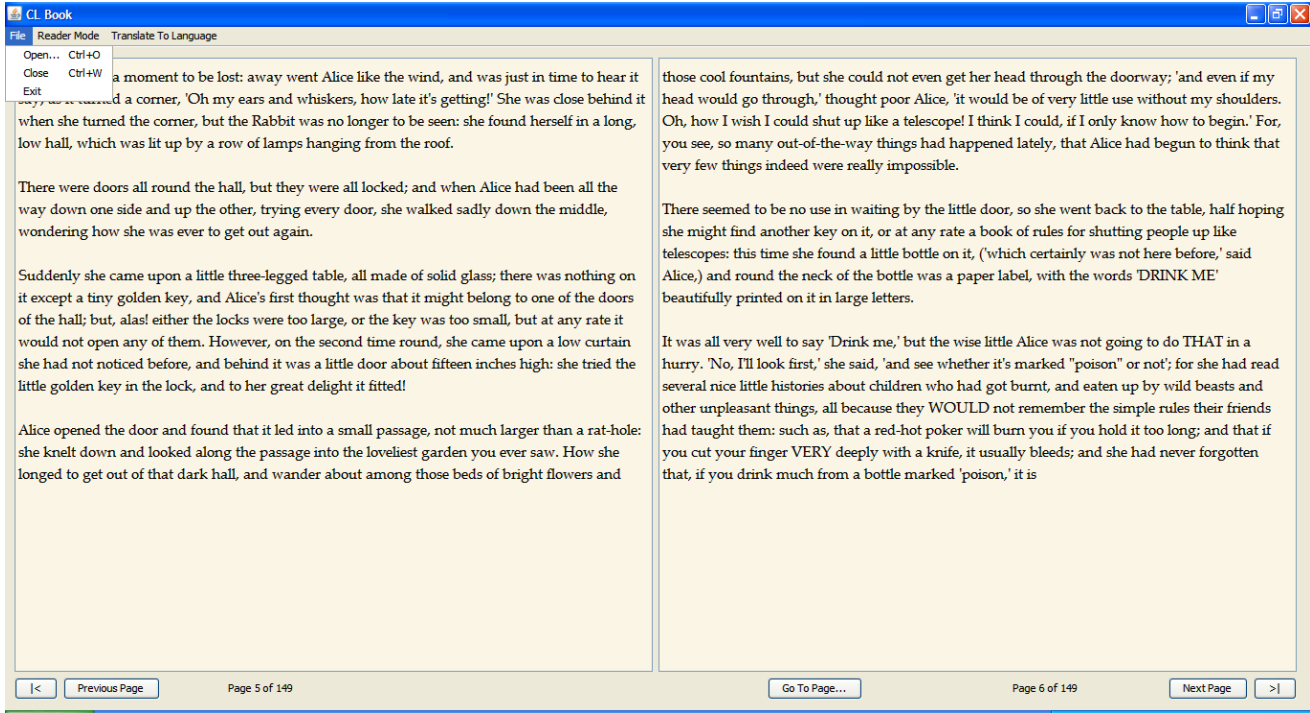


Figure 2. File Menu

The file menu. It allows the user to Open, Close a book, as well as to Exit the application.

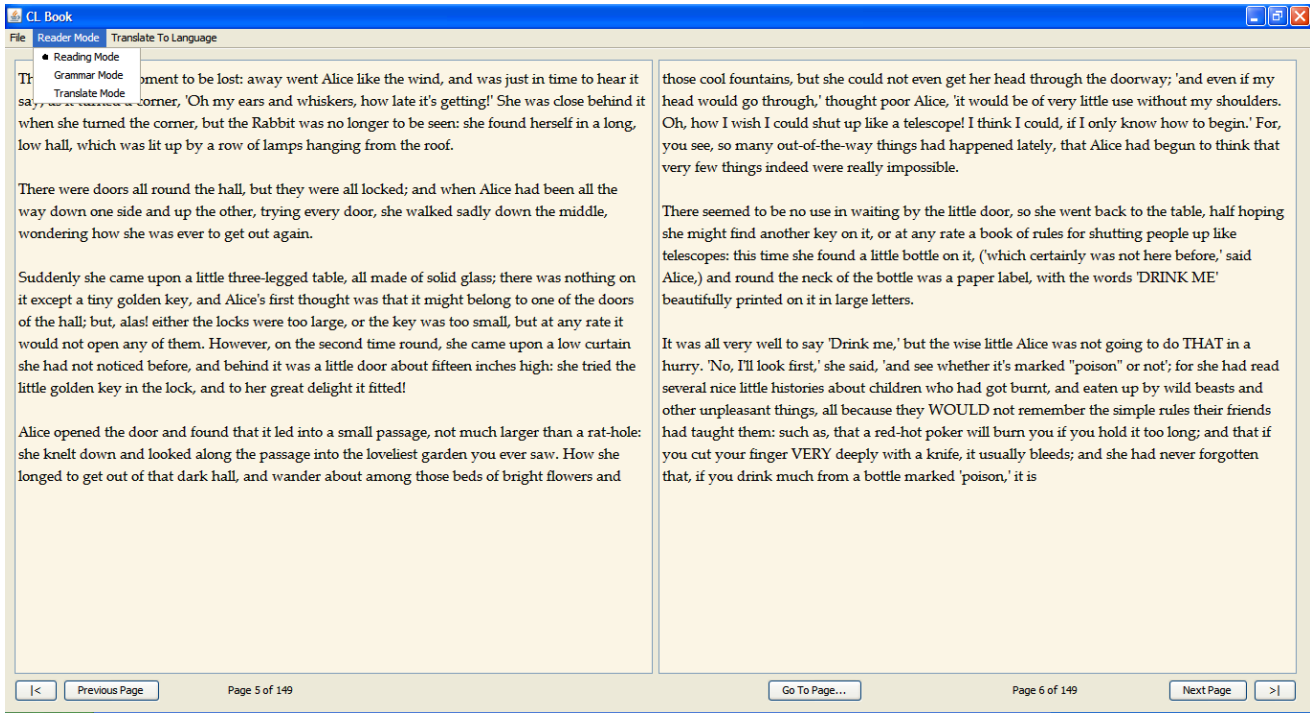


Figure 3. Mode Menu

There are three modes of operation altogether in CL Book: Reading Mode, Grammar Mode, and Translation Mode. This menu allows you switch between them.



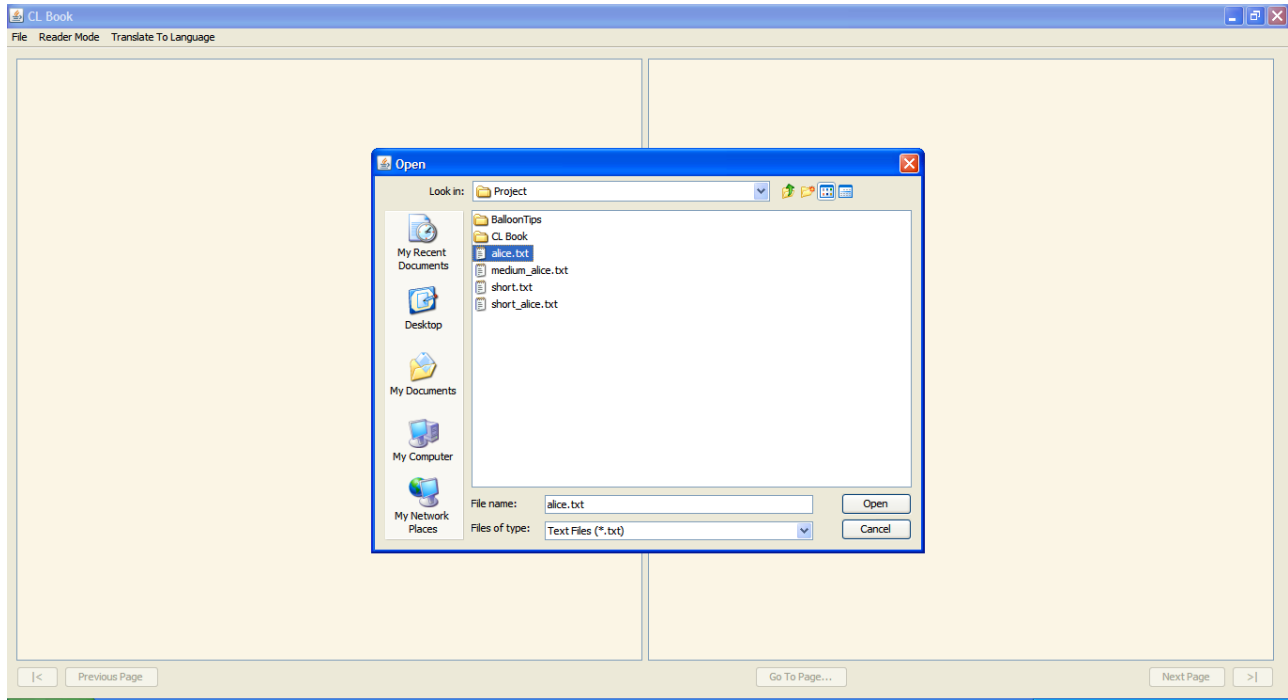


Figure 4. Open Book Dialog

This is the Open book dialog. It is accessible by pressing Ctrl+O or by going under the File menu, and selecting Open. You can choose the book of your choice, in a \*.txt plain text file format.

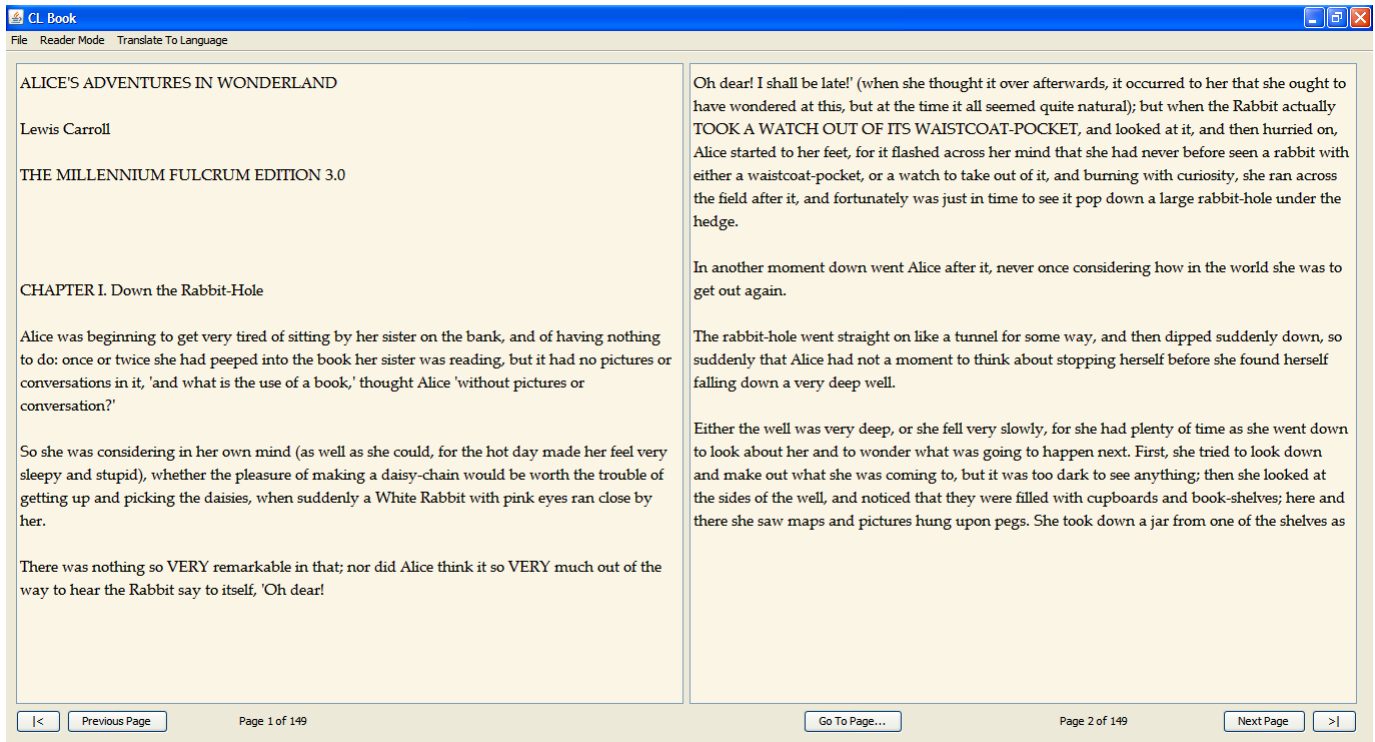


Figure 5. Sample Book Opened (Reading Mode)

This is the default view after opening a book. The mode visible here is the Reading Mode, and it acts most similarly to any standard eBook reader. It displays two pages of the book at a time, and allows the user to read the book as is, without any guides or augmentations.

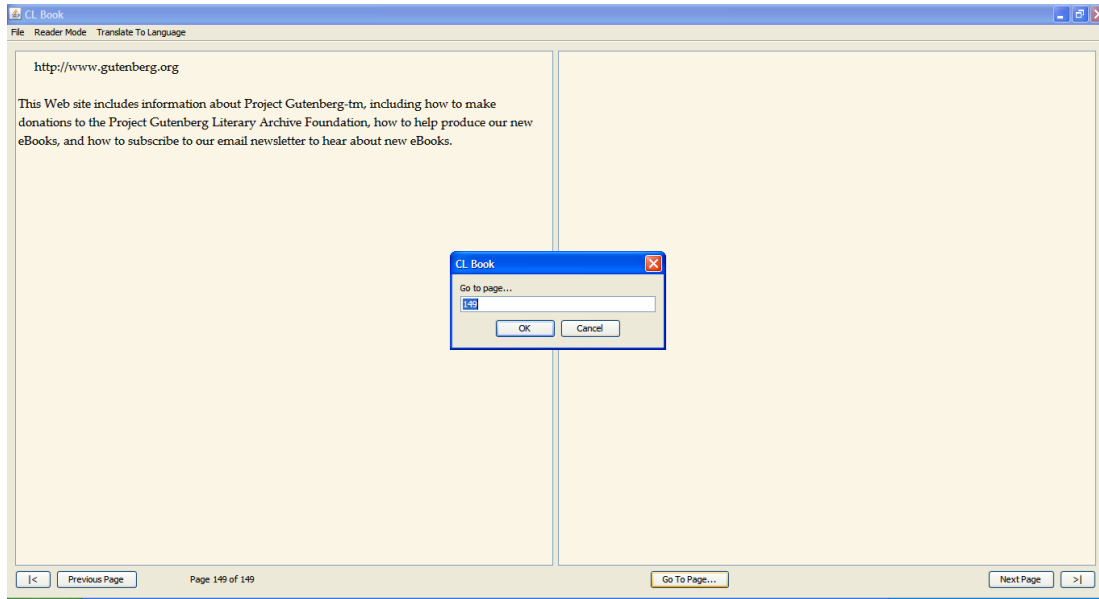


Figure 6. Go To Page Dialog

To complete the eBook reading experience, there are buttons to go to the beginning or the end of the book. There are buttons to go to the previous and next page. There is also a button to go to a specific page, as shown here. Upon clicking, it prompts the user for the page number he or she wishes to jump directly to.

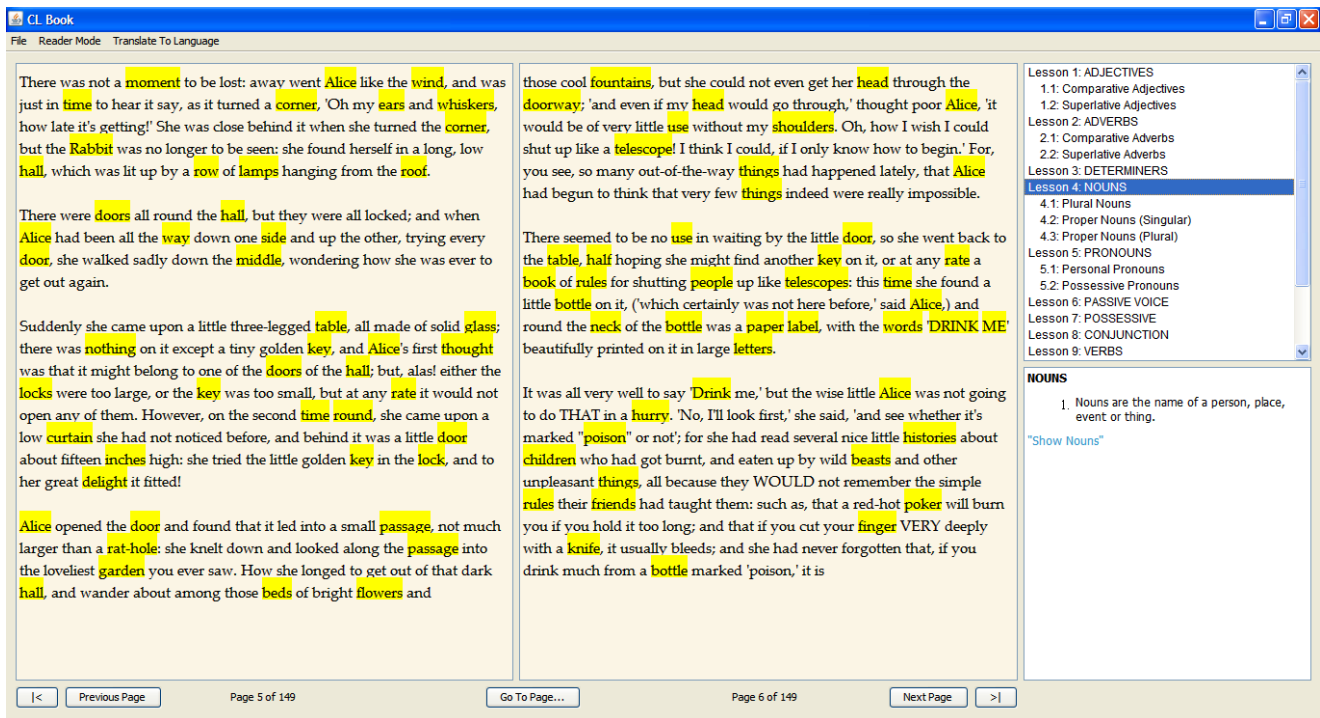


Figure 7. Grammar Mode, Nouns Lesson

The Grammar Mode brings in a sidebar with a list of lessons on the right. There are a number of grammar lessons that help the user identify examples of said lessons within the text they are currently reading. In this screenshot, the Nouns lesson is active, and it highlights the occurrences of nouns.



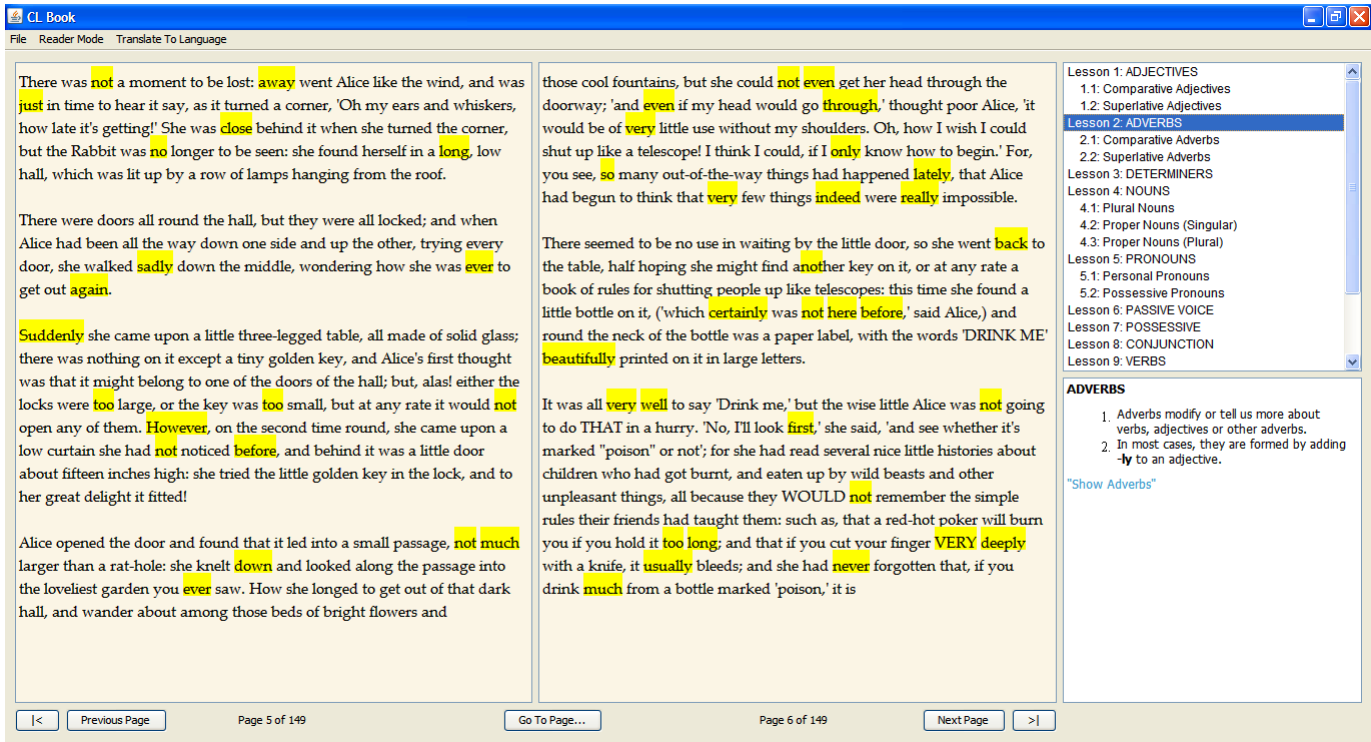


Figure 10. Grammar Mode, Adverbs Lesson

The Adverbs lesson highlights the occurrences of adverbs.

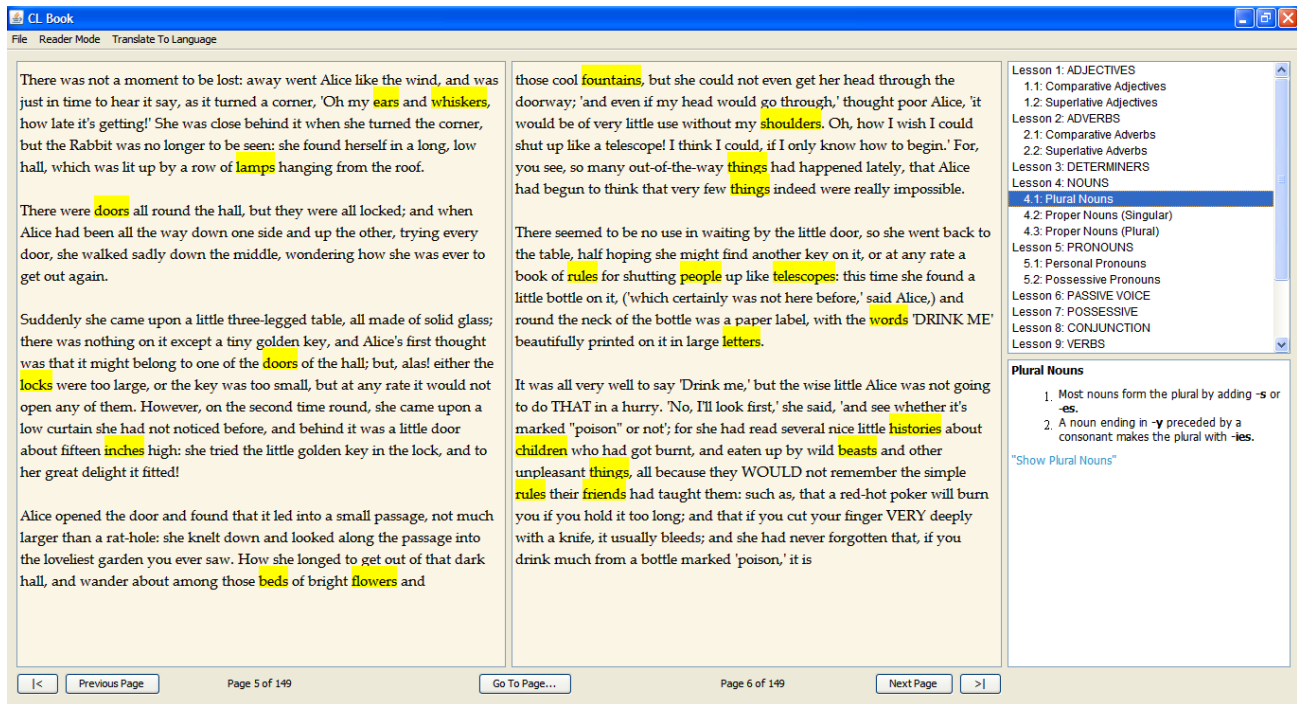


Figure 11. Grammar Mode, Plural Nouns Lesson

This is an example of a sub-lesson, which further expands on a given topic. In the Plural Nouns lesson, only the plural nouns are highlighted.



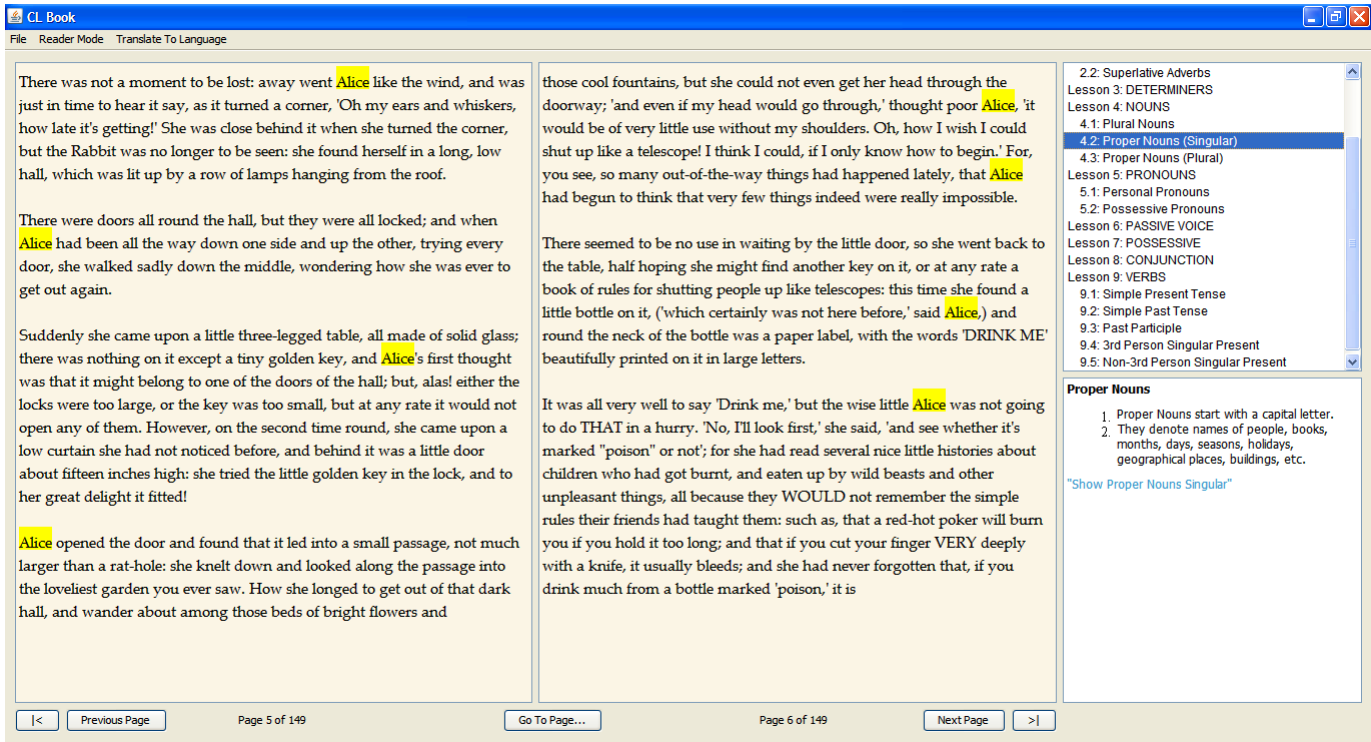


Figure 12. Grammar Mode, Proper Nouns (Singular) Lesson

In the Proper Nouns (Singular) lessons, only the singular proper nouns are highlighted.

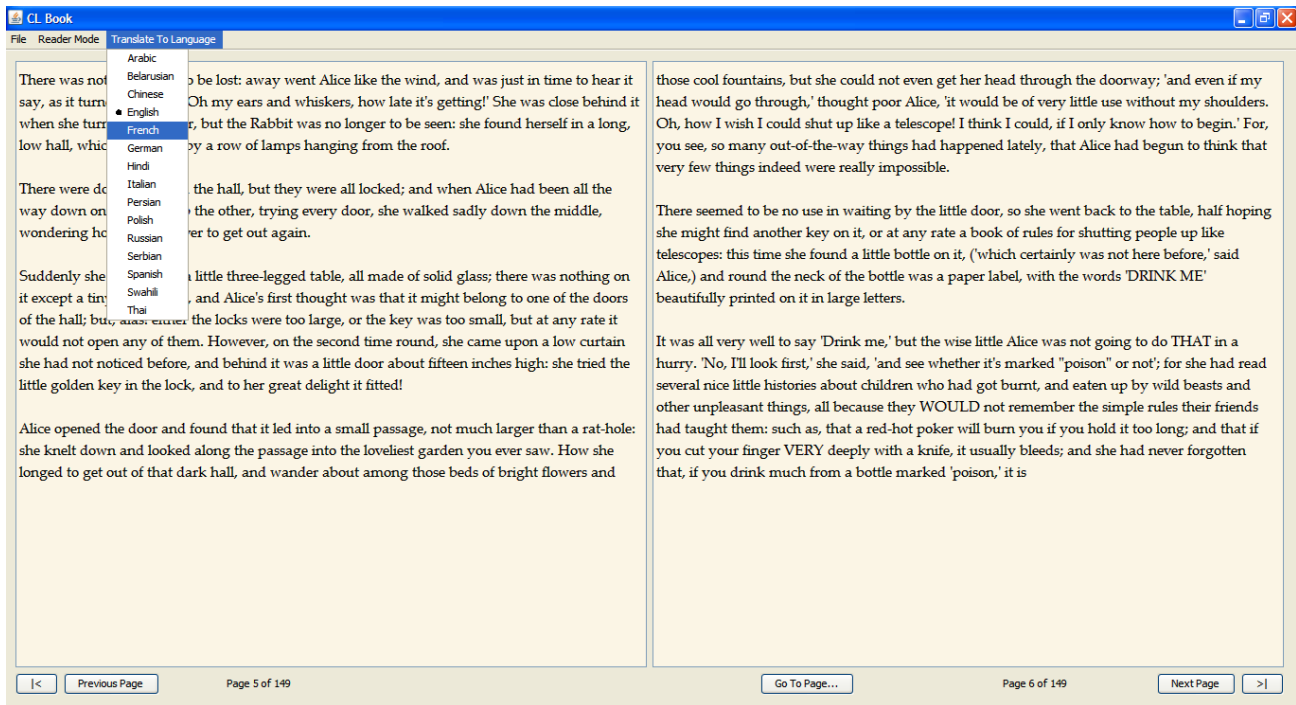


Figure 13. Language Menu

This is the Translate To Language menu, and it allows the user to choose from one of 15 languages. When translation tools are used (shown below), this will be the target language of the translation.

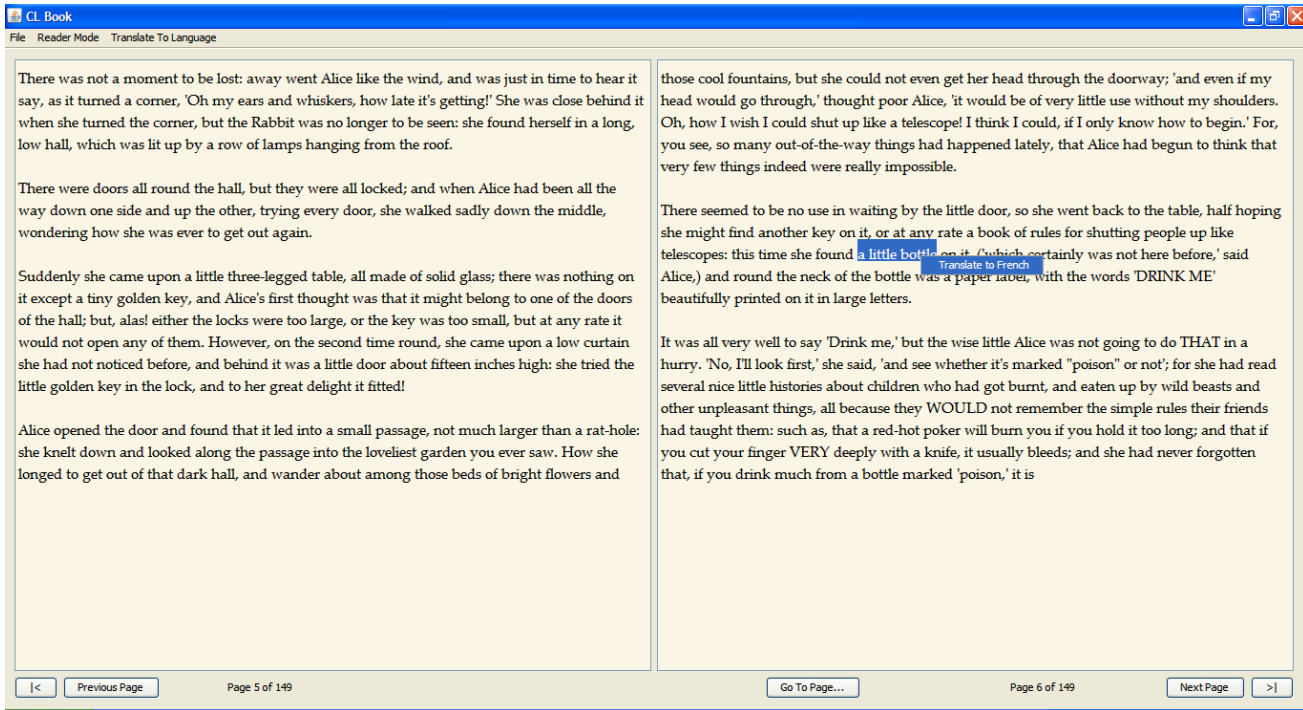


Figure 14. Right Click Context Menu, Quick Translation

This is a translation tool that is always available to the user, no matter which of three modes of operation he or she is currently in. By right clicking on a word, or a selection, a context menu appears and allows you to quickly see a translation of the selected word or sentences.

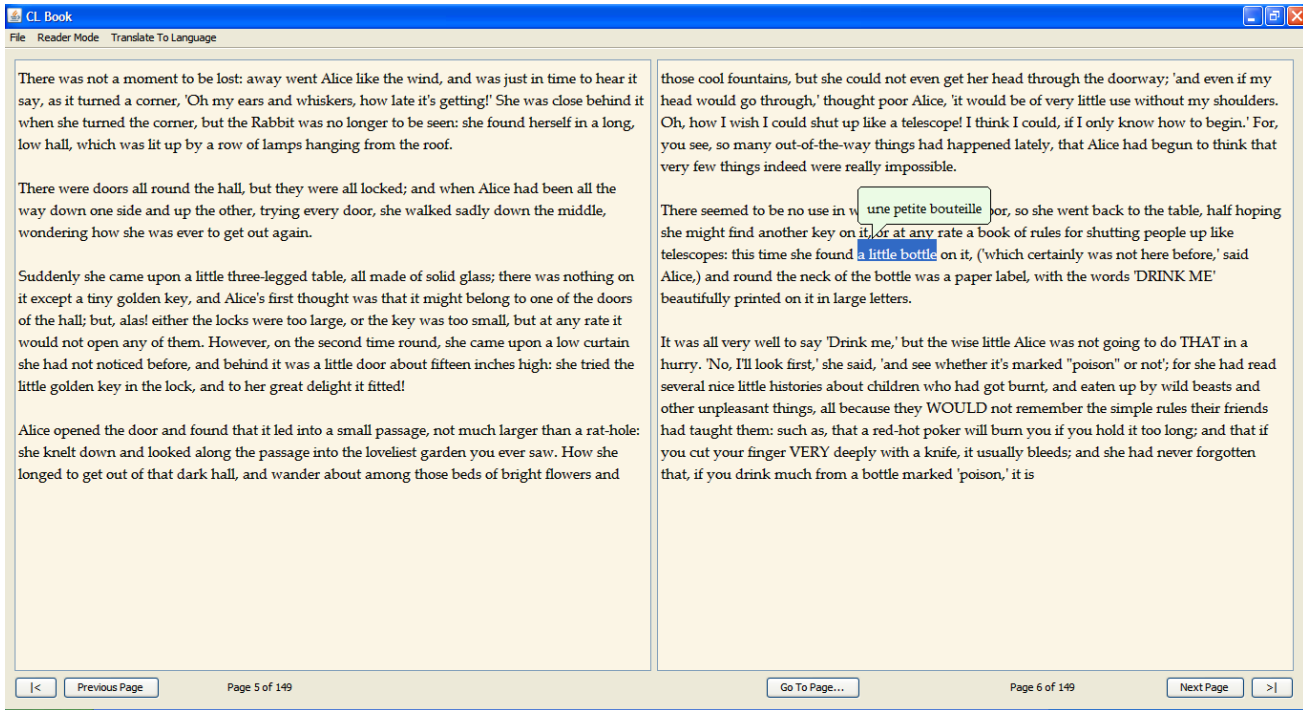


Figure 15. Quick Translation in a Balloon Tooltip

This is how the translated text appears. Clicking anywhere makes the balloon tooltip disappear.

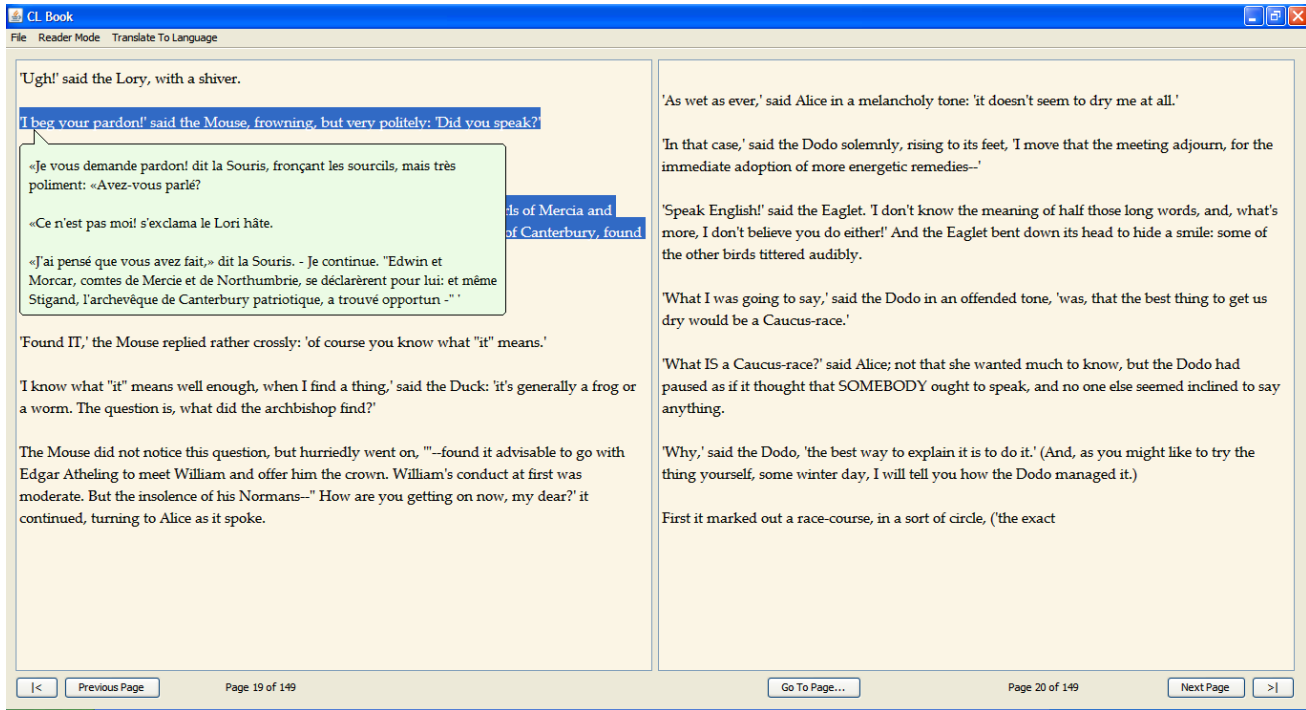


Figure 16. Quick Translation of a Larger Text Section

This translation tool can be used to translate larger blocks of text, while preserving the formatting of the text.

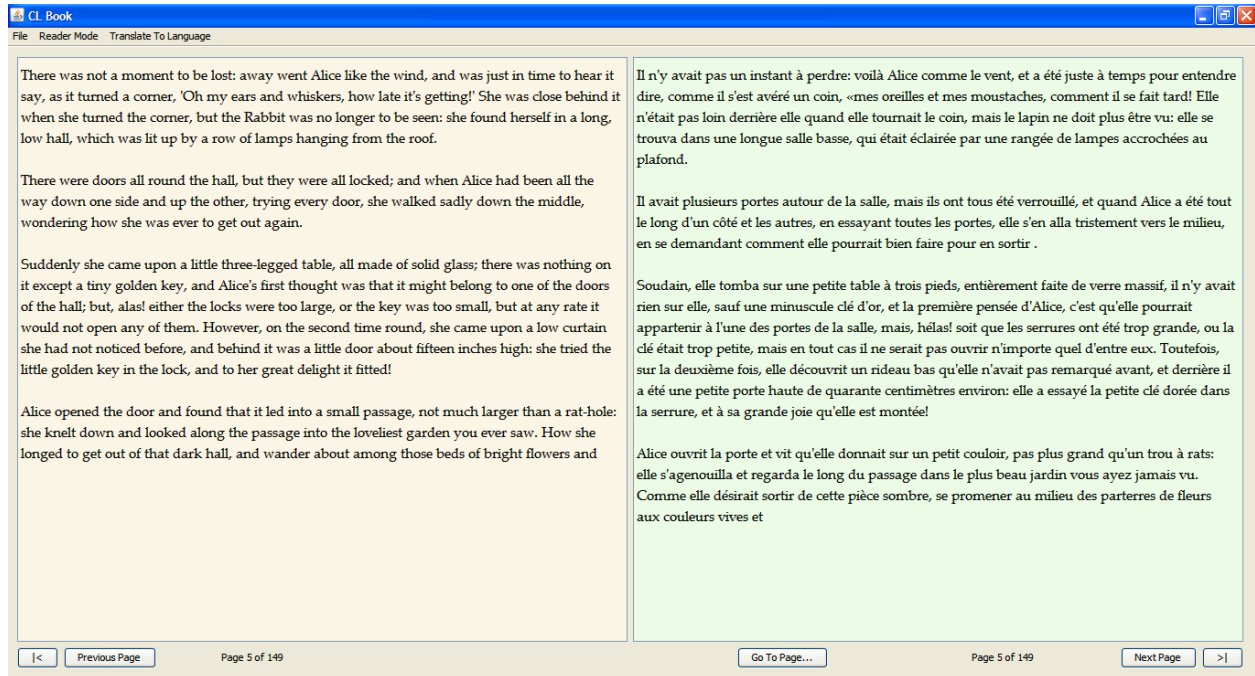


Figure 17. Translate Mode (to French)

The third mode of operation is the Translate Mode. It allows for permanent translation of the given text to one of 15 supported languages. It translates a page at a time, displaying the original page on the left and the translated version on the right. In this example, the English text is translated to French.

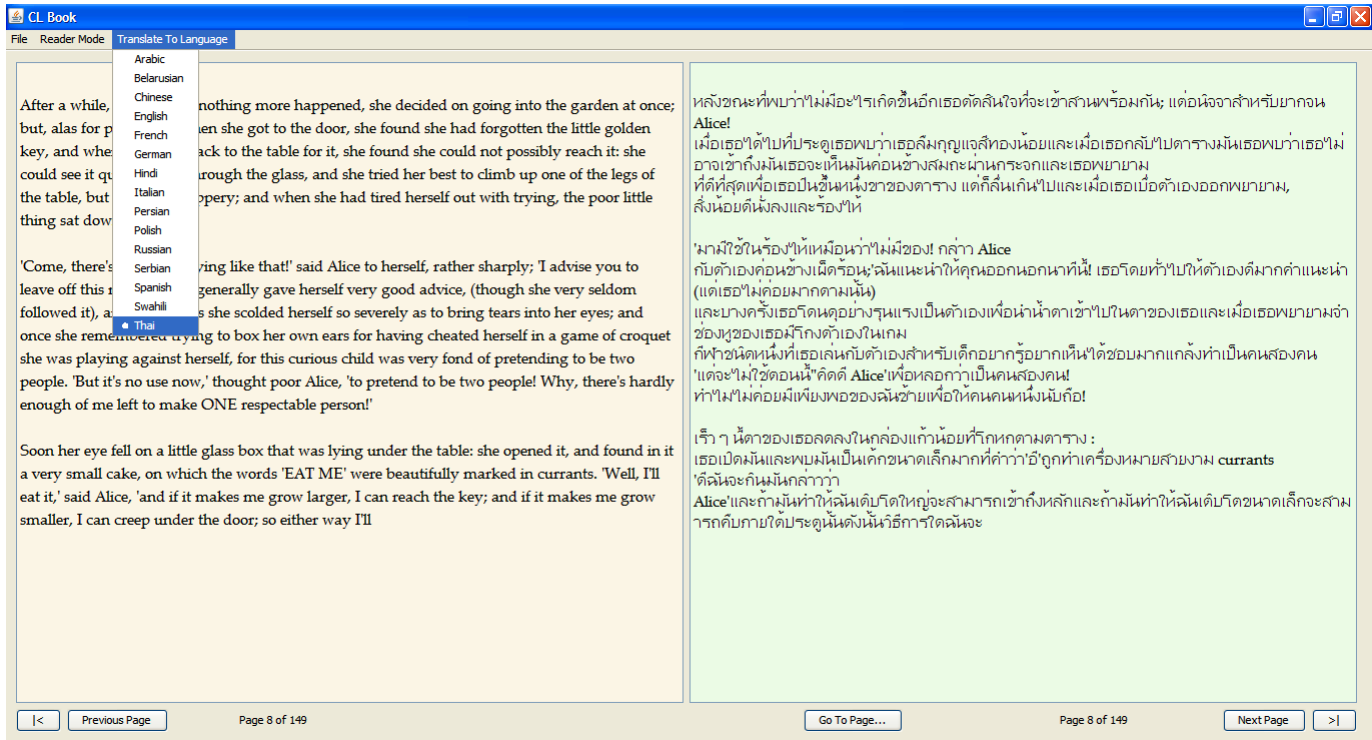


Figure 18. Translate Mode (to Thai)

Here, the same text is translated to Thai.

## 4. Conclusion and Future Work

In this project, we have designed and built a system that allows the user to read books while augmenting their experience using various computational linguistics methods. The user is able to quickly translate any part of the text into one of sixteen languages. The program also allows beginners to learn English grammar by visualizing lessons and examples from their favourite e-book. This puts the examples in context and makes learning intuitive and fun.

We would like to extend our system to include a dictionary as well as voice reader.

**Dictionary bundling** – Continuing with the theme of practically unlimited storage, we can easily imagine a book being bundled with a look-up dictionary that is capable of providing a translation of every word and expression in the text.

**Voice reader** – With a text-to-speech translator, we could have an option where the software automatically reads out the e-book.



## 5. System Manual

### 5.1 Deployment

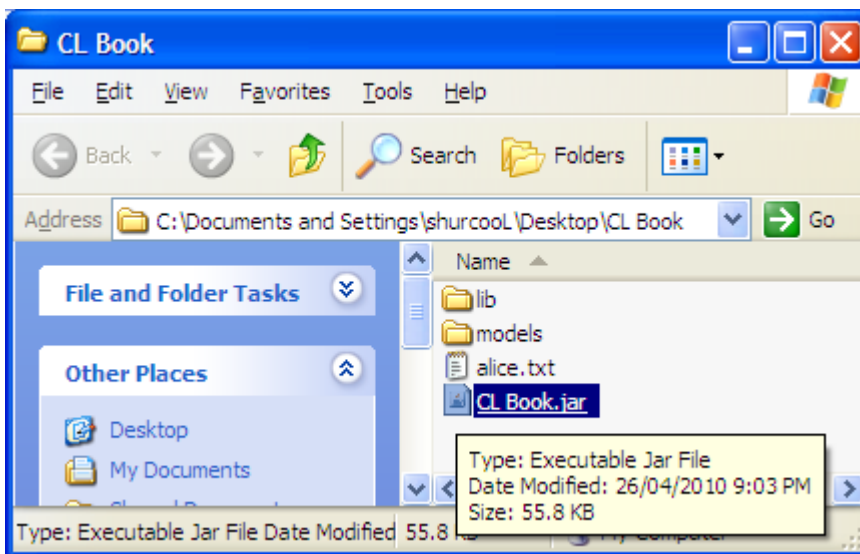
#### 5.1.1. As Java Web-Start application

The program is distributed as a Java Web-Start application accessible over the Internet. Click "Launch" button at the following url <http://www.cse.yorku.ca/~ameeta/CLBook/launch.html> to launch the application.

Requirements: A compatible version of the Java Runtime Environment (JRE) installed on the client machine. The installation of the Java Development Kit (JDK) is not required.

#### 5.1.2. As Java Desktop application

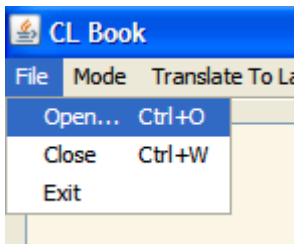
- Please download and extract the "CL\_Book.zip" archive.
- Extract it to any folder and navigate there.
- Double click the CL\_Book.jar file to start the program.



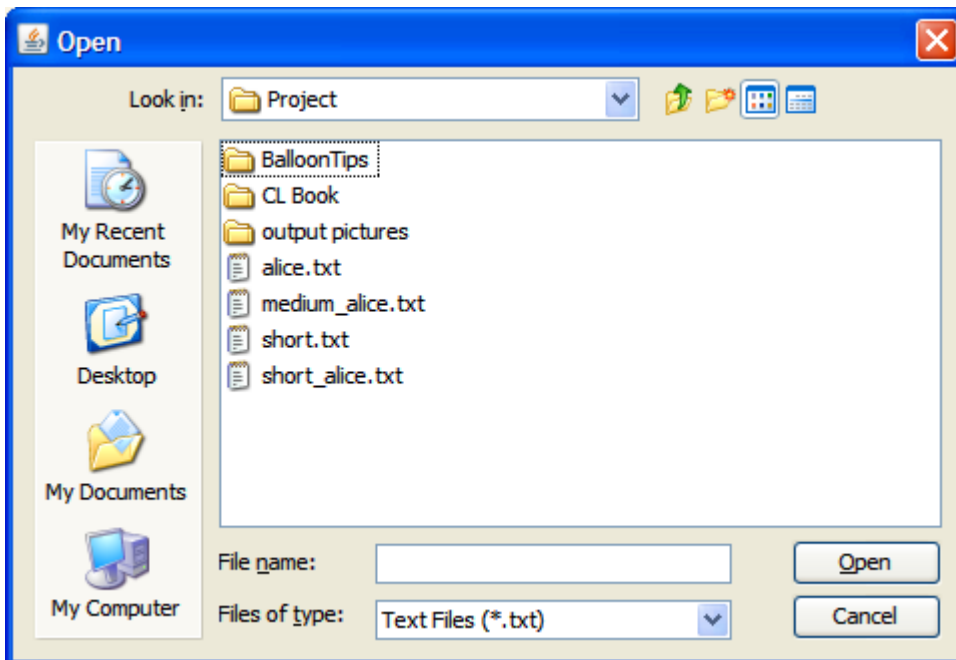
## 5.2 Functionality

### 5.2.1 Opening a book

In order to open a book, select File from the menu bar, and click Open. Alternatively, you can use the Ctrl+O shortcut.



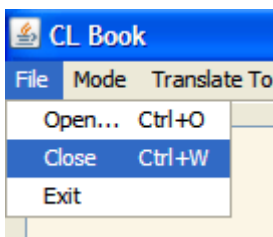
This will bring up an Open dialog.



Navigate to the folder where your eBook resides. Note that the CL Book program only supports books in plain text format at this time. They must have .txt extension. You can choose All Files from "Files of type" if your file has a different extension. Click Open or double-click your book to load it.

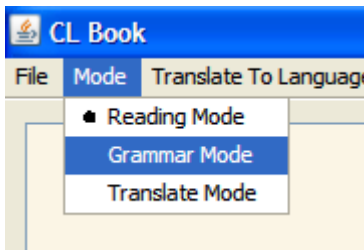
### 5.2.2 Closing a book

To close a book, select File from the menu bar, and click Close. You can also use the Ctrl+W shortcut. Additionally, you can open another book directly.



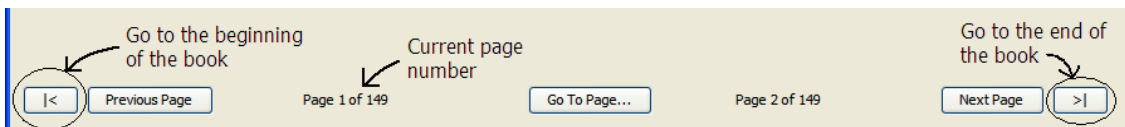
### 5.2.3 Changing mode of operation

To change the mode of operation, select Mode from menu bar. This allows you to choose between the three modes.



#### 5.2.3.1 Using the Reading Mode

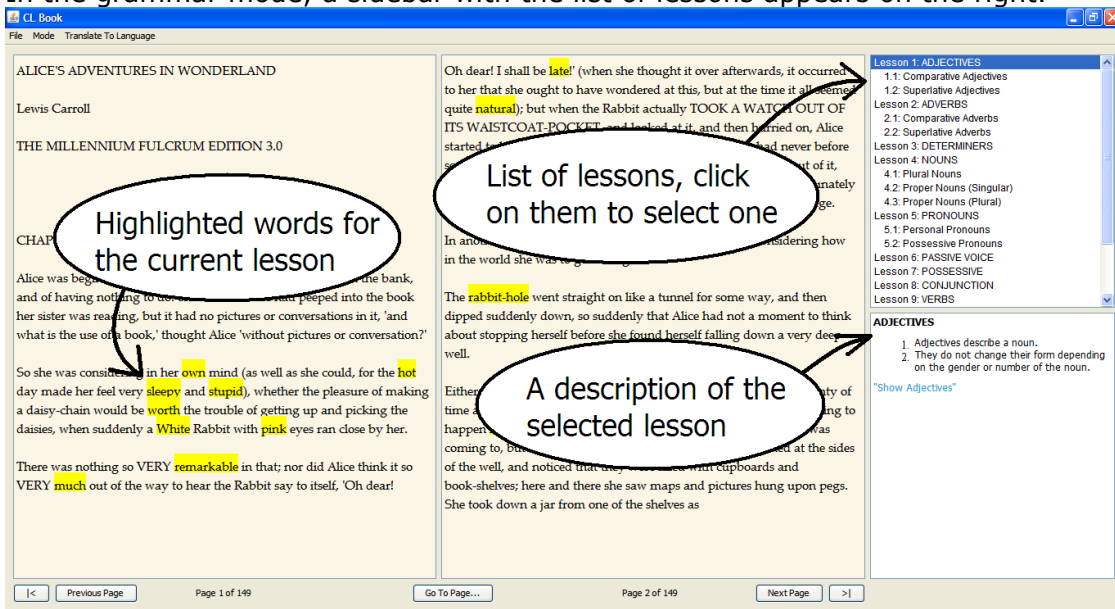
This Mode offers simple eBook reading functionality. The page navigation controls are at the bottom. When a book is loaded, they become enabled.



There are buttons to go to the beginning/end of the book. There are also buttons to go to the next/previous pages. It shows you the current page number underneath each page, and there is a "Go To Page" button in the middle.

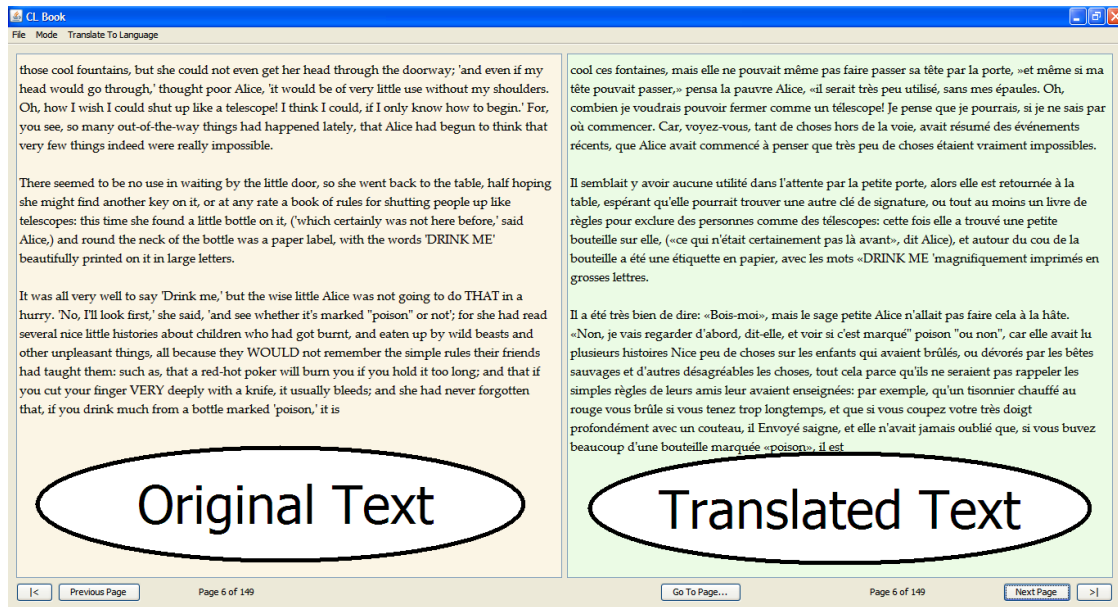
#### 5.2.3.2 Using the Grammar Mode

In the grammar mode, a sidebar with the list of lessons appears on the right.



### 5.2.3.3 Using the Translate Mode

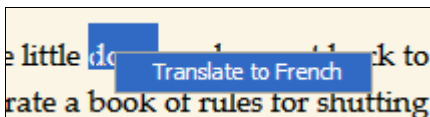
In the Translate Mode, the left hand side shows one page of the original book, and the right hand side shows the translated version.



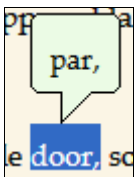
You can control the language to translate to from the menu bar, under the Language To Translate menu.

### 5.2.4 Using the right-click quick translate tool

If you want to quickly translate a word or any amount of text in the reader, you can simply right click on a word or a selection to bring up a context menu.



Click on "Translate to Language" (note that you can control the target language from the menu bar).



A translation appears. Click anywhere on the text again to close the balloon tooltip.

## 6. References

- [1] Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. [Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network](#). In *Proceedings of HLT-NAACL 2003*, pp. 252-259.
- [2] Kristina Toutanova and Christopher D. Manning. 2000. [Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger](#). In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*, pp. 63-70.
- [3] List of grammar topics: <http://www.learnenglish.de/grammarpage.htm>, accessed on Apr 26, 2010
- [4] List of grammar topics: [http://www.edufind.com/english/grammar/grammar\\_topics.php](http://www.edufind.com/english/grammar/grammar_topics.php), accessed on Apr 26, 2010
- [5] [Building a Large Annotated Corpus of English: The Penn Treebank](#), Mitchell P. Marcus, Mary Ann Marcinkiewicz, Beatrice Santorini

## 7. Appendix

### 7.1. The Penn Treebank POS tagset

1. CC Coordinating conjunction
2. CD Cardinal number
3. DT Determiner
4. EX Existential *there*
5. FW Foreign word
6. IN Preposition/subordinating participle conjunction
7. JJ Adjective
8. JJR Adjective, comparative
9. JJS Adjective, superlative
10. LS List item marker
11. MD Modal
12. NN Noun, singular or mass
13. NNS Noun, plural
14. NNP Proper noun, singular
15. NNPS Proper noun, plural
16. PDT Predeterminer
17. POS Possessive ending
18. PRP Personal pronoun
19. PRP\$ Possessive pronoun
20. RB Adverb
21. RBR Adverb, comparative
22. RBS Adverb, superlative
23. RP Particle
24. SYM Symbol (mathematical or scientific)
25. TO *to*
26. UH Interjection
27. VB Verb, base form
28. VBD Verb, past tense
29. VBG Verb, gerund/present
30. VBN Verb, past participle
31. VBP Verb, non-3rd ps. sing. Present
32. VBZ Verb, 3rd ps. sing. present
33. WDT *wh*-determiner
34. WP *wh*-pronoun
35. WP\$ Possessive *wh*-pronoun
36. WRB *wh*-adverb
37. # Pound sign
38. \$ Dollar sign
39. . Sentence-final punctuation

- 40. , Comma
- 41. : Colon, semi-colon
- 42. ( Left bracket character
- 43. ) Right bracket character
- 44. " Straight double quote
- 45. ' Left open single quote
- 46. " Left open double quote
- 47. ' Right close single quote
- 48. " Right close double quote

## 7.2 CL Lessons.txt

### Lesson 1: ADJECTIVES

adjective

```
<font face="Tahoma" size=3>
<strong>ADJECTIVES</strong>
<ol>
<li>Adjectives describe a noun.</li>
<li>They do not change their form depending on the gender or number of the noun.</li>
</ol>
<font color="#3399CC">"Show Adjectives"</font><br>
</font>
```

#### 1.1: Comparative Adjectives

adjectiveComp

```
<font face="Tahoma" size=3>
<strong>Comparative Adjectives</strong>
<ol>
<li>Comparative Adjectives: To show adjective in the comparative form <i>more +
adjective</i>.</li>
</ol>
<font color="#3399CC">"Show Comparative Adjectives"</font><br>
</font>
```

#### 1.2: Superlative Adjectives

adjectiveSuper

```
<font face="Tahoma" size=3>
<strong>Superlative Adjectives</strong>
<ol>
<li>Superlative Adjectives: To show adjective in the superlative form <i>most + adjective</i>.</li>
</ol>
<font color="#3399CC">"Show Superlative Adjectives"</font><br>
</font>
```

### Lesson 2: ADVERBS

adverb

```
<font face="Tahoma" size=3>
<strong>ADVERBS</strong>
<ol>
<li>Adverbs modify or tell us more about verbs, adjectives or other adverbs.</li>
<li>In most cases, they are formed by adding <b>-ly</b> to an adjective.</li>
</ol>
<font color="#3399CC">"Show Adverbs"</font><br>
</font>
```