The glossaries package: a guide for beginners

Nicola L.C. Talbot

2014-03-06

Abstract

This document is a brief guide to the glossaries package for beginners who find the size of the main user manual daunting and, as such, it only covers the basics. For brevity some options to the commands described here are omitted. For a more detailed guide, see the main user manual (glossaries-user.pdf).

Contents

1 Defining Terms

When you use the glossaries package, you need to define glossary entries in the document preamble. These entries could be a word, phrase, acronym

or symbol. They're usually accompanied by a description, which could be a short sentence or an in-depth explanation that spans multiple paragraphs. The simplest method of defining an entry is to use:

```
\newglossaryentry{⟨label⟩}
{
  name={⟨name⟩},
  description={⟨description⟩},
  ⟨other options⟩
}
```

where ⟨*label*⟩ is a unique label that identifies this entry. (Don't include the angle brackets ⟨ ⟩. They just indicate the parts of the code you need to change when you use this command in your document.) As with similar labelling commands, such as `\label` or `\bibitem`, the label should not contain active characters, so just use `a`, ..., `z`, `A`, ..., `Z`, `0`, ..., `9`. You may also be able to use some punctuation characters, unless they have been made active (for example, via babel's shorthand activation.) The ⟨*name*⟩ is the word, phrase or symbol you are defining, and ⟨*description*⟩ is the description.

This command is a "short" command, which means that ⟨*description*⟩ can't contain a paragraph break. If you have a long description, you can instead use:

```
\longnewglossaryentry{⟨label⟩}
{
  name={⟨name⟩},
  ⟨other options⟩
}
{⟨description⟩}
```

Examples:

1. Define the term "set" with the label `set`:

```
\newglossaryentry{set}
{
  name={set},
  description={a collection of objects}
}
```

2. Define the symbol ∅ with the label `emptyset`:

```
\newglossaryentry{emptyset}
{
```

```
  name={\ensuremath{\emptyset}},
  description={the empty set}
}
```

3. Define the phrase "Fish Age" with the label `fishage`:

```
\longnewglossaryentry{fishage}
{name={Fish Age}}
{%
  A common name for the Devonian geologic period
  spanning from the end of the Silurian Period to
  the beginning of the Carboniferous Period.

  This age was known for its remarkable variety of
  fish species.
}
```

   (The percent character discards the end of line character that would otherwise cause an unwanted space to appear at the start of the description.)

4. Take care if the first letter is an extended Latin or non-Latin character (either specified via a command such as `\'e` or explicitly via the inputenc package such as é). This first letter must be placed in a group:

```
\newglossaryentry{elite}
{
  name={{\'e}lite},
  description={select group or class}
}
```

   or

```
\newglossaryentry{elite}
{
  name={{é}lite},
  description={select group or class}
}
```

Acronyms or abbreviations can be defined using

`\newacronym{`⟨*label*⟩`}{`⟨*short*⟩`}{`⟨*long*⟩`}`

where ⟨*label*⟩ is the label (as with the `\newglossaryentry` and the `\longnewglossaryentry` commands), ⟨*short*⟩ is the abbreviation or acronym and ⟨*long*⟩ is the long form. For example:

```
\newacronym{svm}{svm}{support vector machine}
```

This defines a glossary entry with the label `svm`. By default, the ⟨*name*⟩ is set to ⟨*short*⟩ ("svm" in the above example) and the ⟨*description*⟩ is set to ⟨*long*⟩ ("support vector machine" in the above example). If, instead, you want to be able to specify your own description you can do this using the optional argument:

```
\newacronym
 [description={statistical pattern recognition technique}]
 {svm}{svm}{support vector machine}
```

There are other keys you can use when you define an entry. For example, the `name` key used above indicates how the term should appear in the list of entries (glossary). If the term should appear differently when you reference it in the document, you need to use the `text` key as well.

For example:

```
\newglossaryentry{latinalph}
{
  name={Latin Alphabet},
  text={Latin alphabet},
  description={alphabet consisting of the letters
  a, \ldots, z, A, \ldots, Z}
}
```

This will appear in the text as "Latin alphabet" but will be listed in the glossary as "Latin Alphabet".

Another commonly used key is `plural` for specifying the plural of the term. This defaults to the value of the `text` key with an "s" appended, but if this is incorrect, just use the `plural` key to override it:

```
\newglossaryentry{oesophagus}
{
  name={{\oe}sophagus},
  plural={{\oe}sophagi},
  description={canal from mouth to stomach}
}
```

(Remember from earlier that the initial ligature `\oe` needs to be grouped.)

The plural forms for acronyms can be specified using the `longplural` and `shortplural` keys. For example:

```
\newacronym
 [longplural={diagonal matrices}]
 {dm}{DM}{diagonal matrix}
```

If omitted, the defaults are again obtained by appending an "s" to the singular versions.

It's also possible to have both a name and a corresponding symbol. Just use the `name` key for the name and the `symbol` key for the symbol. For example:

```
\newglossaryentry{emptyset}
{
  name={empty set},
  symbol={\ensuremath{\emptyset}},
  description={the set containing no elements}
}
```

2 Using Entries

Once you have defined your entries, as described above, you can reference them in your document. There are a number of commands to do this, but the most common one is:

```
\gls{⟨label⟩}
```

where ⟨*label*⟩ is the label you assigned to the entry when you defined it. For example, `\gls{fishage}` will display "Fish Age" in the text (given the definition from the previous section).

If the entry was defined as an acronym (using `\newacronym` described above), then `\gls` will display the full form the first time it's used and just the short form on subsequent use. For example, `\gls{svm}` will display "support vector machine (svm)" the first time it's used, but the next occurrence of `\gls{svm}` will just display "svm".

If you want the plural form, you can use:

```
\glspl{⟨label⟩}
```

instead of `\gls{⟨label⟩}`. For example, `\glspl{set}` displays "sets".

If the term appears at the start of a sentence, you can convert the first letter to upper case using:

```
\Gls{⟨label⟩}
```

for the singular form or

```
\Glspl{⟨label⟩}
```

for the plural form. For example:

```
\Glspl{set} are collections.
```

produces "Sets are collections".

If you've specified a symbol using the `symbol` key, you can display it using:

\glssymbol{⟨*label*⟩}

3 Acronyms

Recall from above, the first time you use an acronym with \gls, it's full form is displayed but subsequent uses display only the short form. By default, the first use displays ⟨*long*⟩ (⟨*short*⟩). That is, the long form is displayed followed by the short form in parentheses. You can change this first-use format with:

\setacronymstyle{⟨*style name*⟩}

(This must be used before you start defining your acronyms with \newacronym.) There are a number of predefined styles listed in Section 13.1.1 in the main glossaries user manual. Here are a few examples:

1. ⟨*short*⟩ (⟨*long*⟩)

   ```
   \setacronymstyle{short-long}
   ```

   This displays the short form followed by the long form in parentheses.

2. ⟨*long*⟩ (\textsc{⟨*short*⟩})

   ```
   \setacronymstyle{long-short-sc}
   ```

   This is like the default style but the short form is displayed in small caps. Remember that when you use \textsc{⟨*text*⟩} to generate small capitals, you must specify ⟨*text*⟩ in lower case, so the short form of the acronym should be defined in lower case. For example:

   ```
   \newacronym{svm}{svm}{support vector machine}
   ```

3. ⟨*long*⟩ (\textsmaller{⟨*short*⟩})

   ```
   \setacronymstyle{long-short-sm}
   ```

   This is similar to the previous style but uses \textsmaller to format the short form. Remember to load the relsize package, which defines \textsmaller, if you want to use this style. The short form now needs to be defined in upper case:

   ```
   \newacronym{svm}{SVM}{support vector machine}
   ```

4 Displaying a List of Entries

Suppose you now want to display a list of all the entries you've referenced in your document. This is where things start to get complicated and a lot of new users get bewildered. You have three options:

Option 1:

This is the simplest option but it's slow and if you want a sorted list, it doesn't work for non-Latin alphabets.

1. Add `\makenoidxglossaries` to your preamble (before you start defining your entries, as described in Section 1).
2. Put

   ```
   \printnoidxglossary[sort=⟨order⟩,⟨other options⟩]
   ```

   where you want your list of entries to appear. The sort ⟨*order*⟩ may be one of: `word` (word ordering), `letter` (letter ordering), `case` (case-sensitive letter ordering), `def` (in order of definition) or `use` (in order of use).
3. Run LaTeX twice on your document. (As you would do to make a table of contents appear.) For example, click twice on the "typeset" or "build" or "PDFLaTeX" button in your editor.

Option 2:

This option uses an application called `makeindex` to sort the entries. This application comes with all modern TeX distributions, but it's hard-coded for the non-extended Latin alphabet. This process involves making LaTeX write the glossary information to a temporary file which `makeindex` reads. Then `makeindex` writes a new file containing the code to typeset the glossary. LaTeX then reads this file on the next run.

1. Add `\makeglossaries` to your preamble (before you start defining your entries).
2. Put

   ```
   \printglossary[⟨options⟩]
   ```

   where you want your list of entries (glossary) to appear.
3. Run LaTeX on your document. This creates files with the extensions `.glo` and `.ist` (for example, if your LaTeX document

is called `myDoc.tex`, then you'll have two extra files called `myDoc.glo` and `myDoc.ist`). If you look at your document at this point, you won't see the glossary as it hasn't been created yet.

4. Run `makeindex` with the `.glo` file as the input file and the `.ist` file as the style so that it creates an output file with the extension `.gls`. If you have access to a terminal or a command prompt (for example, the MSDOS command prompt for Windows users or the bash console for Unix-like users) then you need to run the command:

   ```
   makeindex -s myDoc.ist -o myDoc.gls myDoc.glo
   ```

   (Replace `myDoc` with the base name of your LaTeX document file. Avoid spaces in the file name.) If you don't know how to use the command prompt, then you can probably access `makeindex` via your text editor, but each editor has a different method of doing this, so I can't give a general description. You will have to check your editor's manual.

   The default sort is word order ("sea lion" comes before "seal"). If you want letter ordering you need to add the `-l` switch:

   ```
   makeindex -l -s myDoc.ist -o myDoc.gls myDoc.glo
   ```

5. Once you have successfully completed the previous step, you can now run LaTeX on your document again.

Option 3:

This option uses an application called `xindy` to sort the entries. This application is more flexible than `makeindex` and is able to sort extended Latin or non-Latin alphabets. It comes with TeX Live but not with MiKTeX. Since `xindy` is a Perl script, if you are using MiKTeX you will not only need to install `xindy`, you will also need to install Perl. In a similar way to Option 2, this option involves making LaTeX write the glossary information to a temporary file which `xindy` reads. Then `xindy` writes a new file containing the code to typeset the glossary. LaTeX then reads this file on the next run.

1. Add the `xindy` option to the glossaries package option list:

   ```
   \usepackage[xindy]{glossaries}
   ```

2. Add `\makeglossaries` to your preamble (before you start defining your entries).

3. Put

   ```
   \printglossary[⟨options⟩]
   ```

where you want your list of entries (glossary) to appear.

4. Run LaTeX on your document. This creates files with the extensions `.glo` and `.xdy` (for example, if your LaTeX document is called `myDoc.tex`, then you'll have two extra files called `myDoc.glo` and `myDoc.xdy`). If you look at your document at this point, you won't see the glossary as it hasn't been created yet.

5. Run `xindy` with the `.glo` file as the input file and the `.xdy` file as a module so that it creates an output file with the extension `.gls`. You also need to set the language name and input encoding. If you have access to a terminal or a command prompt (for example, the MSDOS command prompt for Windows users or the bash console for Unix-like users) then you need to run the command (all on one line):

   ```
   xindy  -L english -C utf8 -I xindy -M myDoc
   -t myDoc.glg -o myDoc.gls myDoc.glo
   ```

   (Replace `myDoc` with the base name of your LaTeX document file. Avoid spaces in the file name. If necessary, also replace `english` with the name of your language and `utf8` with your input encoding.) If you don't know how to use the command prompt, then you can probably access `xindy` via your text editor, but each editor has a different method of doing this, so I can't give a general description. You will have to check your editor's manual.

   The default sort is word order ("sea lion" comes before "seal"). If you want letter ordering you need to add the `order=letter` package option:

   ```
   \usepackage[xindy,order=letter]{glossaries}
   ```

6. Once you have successfully completed the previous step, you can now run LaTeX on your document again.

For Options 2 and 3, it can be difficult to remember all the parameters required for `makeindex` or `xindy`, so the glossaries package provides a script called `makeglossaries` that reads the `.aux` file to determine what settings you have used and will then run `makeindex` or `xindy`. Again, this is a command line application and can be run in a terminal or command prompt. For example, if your LaTeX document is in the file `myDoc.tex`, then run:

```
makeglossaries myDoc
```

(Replace `myDoc` with the base name of your LaTeX document file. Avoid spaces in the file name.) If you don't know how to use the command

prompt, you can probably access `makeglossaries` via your text editor. Check your editor's manual for advice. If you are using `arara` then you can just use the directives:

```
arara: pdflatex
arara: makeglossaries
arara: pdflatex
```

The `makeglossaries` script is written in Perl, so you need a Perl interpreter installed. If you are using a Unix-like operating system then you most likely have one installed. If you are using Windows with the TEX Live distribution, then you can use the Perl interpreter that comes with TEX Live. If you are using Windows and MiKTEX then you need to install a Perl distribution for Windows. If you are using Option 3, then you need to do this anyway as `xindy` is also written in Perl. If you are using Option 2 and can't work out how to install Perl (or for some reason don't want to install it) then just use `makeindex` directly, as described above.

When sorting the entries, the string comparisons are made according to each entry's `sort` key. If this is omitted, the `name` key is used. For example, recall the earlier definition:

```
\newglossaryentry{elite}
{
  name={{\'e}lite},
  description={select group or class}
}
```

No `sort` key was used, so it's set to the same as the `name` key: `{\'e}lite`. How this is interpreted depends on which option you have used:

Option 1: By default, the accent command will be stripped so the sort value will be `elite`. This will put the entry in the "E" letter group. If you use the sanitizesort=true package option, the sort value will be interpreted as the sequence of characters: `\ ' e l i t` and `e`. This will place this entry before the "A" letter group since it starts with a symbol.

Option 2: The sort key will be interpreted the sequence of characters: `{ \ ' e } l i t` and `e`. The first character is an opening curly brace `{` so `makeindex` will put this entry in the "symbols" group.

Option 3: `xindy` disregards LATEX commands so it sorts on `elite`, which puts this entry in the "E" group.

If the inputenc package is used and the entry is defined as:

```
\newglossaryentry{elite}
{
```

```
  name={{é}lite},
  description={select group or class}
}
```

then:

Option 1: By default the sort value will be interpreted as `elite` so the entry will be put in the "E" letter group. If you use the sanitizesort=true package option, the sort value will be interpreted as `élite` where é has been sanitized (so it's no longer an active character) which will put this entry before the "A" letter group.

Option 2: `makeindex` doesn't recognise `é` as a letter so it will be put in the symbols group.

Option 3: `xindy` will correctly recognise the sort value `élite` and will place it in whatever letter group is appropriate for the given language setting. (In English, this would just be the "E" letter group.)

Therefore if you have extended Latin or non-Latin characters, your best option is to use `xindy` (Option 3) with the inputenc package. If you use `makeindex` (Option 2) you need to specify the `sort` key like this:

```
\newglossaryentry{elite}
{
  name={{\'e}lite},
  sort={elite},
  description={select group or class}
}
```

If you use Option 1, you may or may not need to use the `sort` key, but you will need to be careful about fragile commands in the `name` key if you don't set the `sort` key.

Table 1 summarises the pros and cons of three options described above.

5 Customising the Glossary

The default glossary style uses the `description` environment to display the entry list. Each entry name is set in the optional argument of `\item` which means that it will typically be displayed in bold. You can switch to medium weight by redefining `\glsnamefont`:

```
\renewcommand*{\glsnamefont}[1]{\textmd{#1}}
```

By default, a full stop is appended to the description. To prevent this from happening use the `nopostdot` package option:

```
\usepackage[nopostdot]{glossaries}
```

Table 1: Glossary Options: Pros and Cons

	Option 1	Option 2	Option 3
Requires an external application?	✘	✔	✔
Requires Perl?	✘	✘	✔
Can sort extended Latin or non-Latin alphabets?	✘†	✘	✔
Efficient sort algorithm?	✘	✔	✔
Can form ranges in the location lists?	✘	✔	✔
Can have non-standard locations?	✔	✘	✔
`\newglossaryentry` restricted to preamble?	✔	✘	✘

† Strips standard LaTeX accents so, for example, `\AA` is treated the same as A.

By default, a location list is displayed for each entry. This refers to the document locations (for example, the page number) where the entry has been referenced. If you use Options 2 or 3 described in Section 4 location ranges will be compressed. For example, if an entry was used on pages 1, 2 and 3, with Options 2 or 3 the location list will appear as 1–3, but with Option 1 it will appear as 1, 2, 3. If you don't want the locations displayed you can hide them using the `nonumberlist` package option:

```
\usepackage[nonumberlist]{glossaries}
```

Entries are grouped according to the first letter of each entry's `sort` key. By default a vertical gap is placed between letter groups. You can suppress this with the `nogroupskip` package option:

```
\usepackage[nogroupskip]{glossaries}
```

If the default style doesn't suit your document, you can change the style using:

`\setglossarystyle{`⟨*style name*⟩`}`

There are a number of predefined styles. Glossaries can vary from a list of symbols with a terse description to a list of words or phrases with de-

scriptions that span multiple paragraphs, so there's no "one style fits all" solution. You need to choose a style that suits your document.

Examples:

1. You have entries where the name is a symbol and the description is a brief phrase or short sentence. Try one of the "mcol" styles defined in the glossary-mcols package. For example:

   ```
   \usepackage[nogroupskip,nopostdot]{glossaries}
   \usepackage{glossary-mcols}
   \setglossarystyle{mcolindex}
   ```

2. You have entries where the name is a word or phrase and the description spans multiple paragraphs. Try one of the "altlist" styles. For example:

   ```
   \usepackage[nopostdot]{glossaries}
   \setglossarystyle{altlist}
   ```

3. You have entries where the name is a single word, the description is brief, and an associated symbol has been set. Use one of the styles that display the symbol (not all of them do). For example, one of the tabular styles:

   ```
   \usepackage[nopostdot,nonumberlist]{glossaries}
   \setglossarystyle{long4col}
   ```

   or one of the "tree" styles:

   ```
   \usepackage[nopostdot,nonumberlist]{glossaries}
   \setglossarystyle{tree}
   ```

If your glossary consists of a list of acronyms and you also want to specify a description as well as the long form, then you need to use an acronym style that will suit the glossary style. For example, use the `long-short-desc` acronym style:

```
\setacronymstyle{long-short-desc}
```

Define the acronyms with a description:

```
\newacronym
 [description={statistical pattern recognition technique}]
 {svm}{svm}{support vector machine}
```

Choose a glossary style that suits wide entry names:

```
\setglossarystyle{altlist}
```

6 Multiple Glossaries

The glossaries package predefines a default `main` glossary. When you define an entry (using one of the commands described in Section 1), that entry is automatically assigned to the default glossary, unless you indicate otherwise using the `type` key. However you first need to make sure the desired glossary has been defined. This is done using:

`\newglossary[`⟨*glg*⟩`]{`⟨*label*⟩`}{`⟨*gls*⟩`}{`⟨*glo*⟩`}{`⟨*title*⟩`}`

The ⟨*label*⟩ is a label that uniquely identifies this new glossary. As with other types of identifying labels, be careful not to use active characters in ⟨*label*⟩. The final argument ⟨*title*⟩ is the section or chapter heading used by `\printglossary` or `\printnoidxglossary`. The other arguments indicate the file extensions used by `makeindex`/`xindy` (described in Section 4). If you use Option 1 described above, the ⟨*glg*⟩, ⟨*gls*⟩ and ⟨*glo*⟩ arguments are ignored. In the case of Options 2 or 3, all glossary definitions must come before `\makeglossaries`.

Since it's quite common for documents to have both a list of terms and a list of acronyms, the glossaries package provides the package option `acronyms`, which is a convenient shortcut for

```
\newglossary[alg]{acronym}{acr}{acn}{\acronymname}
```

It also changes the behaviour of `\newacronym` so that acronyms are automatically put in the list of acronyms instead of the main glossary.

For example, suppose you want a main glossary for terms, a list of acronyms and a list of notation:

```
\usepackage[acronyms]{glossaries}
\newglossary[nlg]{notation}{not}{ntn}{Notation}
```

After `\makeglossaries` (or `\makenoidxglossaries`) you can define the entries. For example:

```
\newglossaryentry{gls:set}
{% This entry goes in the 'main' glossary
  name=set,
  description={A collection of distinct objects}
}

 This entry goes in the 'acronym' glossary:
\newacronym{svm}{svm}{support vector machine}

\newglossaryentry{not:set}
{% This entry goes in the 'notation' glossary:
  type=notation,
  name={\ensuremath{\mathcal{S}}},
```

```
  description={A set},
  sort={S}}
```

or if you don't like using \ensuremath:

```
\newglossaryentry{not:set}
{% This entry goes in the 'notation' glossary:
  type=notation,
  name={$\mathcal{S}$},
  text={\mathcal{S}},
  description={A set},
  sort={S}}
```

Each glossary is displayed using:

```
\printnoidxglossary[type=⟨type⟩]
```

(Option 1) or

```
\printglossary[type=⟨type⟩]
```

(Options 2 and 3). Where ⟨*type*⟩ is the glossary label. If the type is omitted the default main glossary is assumed.

There's a convenient shortcut that will display all the defined glossaries:

```
\printnoidxglossaries
```

(Option 1) or

```
\printglossaries
```

(Options 2 and 3).

If you use Option 1, you don't need to do anything else. If you use Options 2 or 3 with the makeglossaries Perl script, you similarly don't need to do anything else. If you use Options 2 or 3 without the makeglossaries Perl script then you need to make sure you run makeindex/xindy *for each defined glossary*. The ⟨*gls*⟩ and ⟨*glo*⟩ arguments of \newglossary specify the file extensions to use instead of .gls and .glo. The optional argument ⟨*glg*⟩ is the file extension for the transcript file. This should be different for each glossary in case you need to check for makeindex/xindy errors or warnings if things go wrong.

For example, suppose you have three glossaries in your document (main, acronym and notation), specified using:

```
\usepackage[acronyms]{glossaries}
\newglossary[nlg]{notation}{not}{ntn}{Notation}
```

Then (assuming your LaTeX document is in a file called `myDoc.tex`):

Option 2:

You need to run `makeindex` three times:

```
makeindex -t myDoc.glg -s myDoc.ist -o myDoc.gls myDoc.glo
makeindex -t myDoc.alg -s myDoc.ist -o myDoc.acr myDoc.acn
makeindex -t myDoc.nlg -s myDoc.ist -o myDoc.not myDoc.ntn
```

Option 3:

You need to run `xindy` three times (be careful not to insert line breaks where the line has wrapped.)

```
xindy  -L english -C utf8 -I xindy -M myDoc -t myDoc.glg
-o myDoc.gls myDoc.glo
xindy  -L english -C utf8 -I xindy -M myDoc -t myDoc.alg
-o myDoc.acr myDoc.acn
xindy  -L english -C utf8 -I xindy -M myDoc -t myDoc.nlg
-o myDoc.not myDoc.ntn
```

7 glossaries and hyperref

Take care if you use the glossaries package with hyperref. Contrary to the usual advice that hyperref should be loaded last, glossaries must be loaded *after* hyperref:

```
\usepackage[colorlinks]{hyperref}
\usepackage{glossaries}
```

If you use hyperref make sure you use PDFLaTeX rather than the LaTeX to DVI engine. The DVI format can't break hyperlinks across a line so long glossary entries (such as the full form of acronyms) won't line wrap with the DVI engine. Also, hyperlinks in sub- or superscripts aren't correctly sized with the DVI format.

By default, if the hyperref package has been loaded, commands like `\gls` will form a hyperlink to the relevant entry in the glossary. If you don't want this to happen for *all* your glossaries, then use

```
\glsdisablehyper
```

If you want hyperlinks suppressed for entries in specific glossaries, then use the `nohypertypes` package option. For example, if you don't want hyperlinks for entries in the `acronym` and `notation` glossaries but you do want them for entries in the `main` glossary, then do:

```
\usepackage[colorlinks]{hyperref}
\usepackage[acronym,nohypertypes={acronym,notation}]{glossaries}
\newglossary[nlg]{notation}{not}{ntn}{Notation}
```

If you want the hyperlinks suppressed the first time an entry is used, but you want hyperlinks for subsequence references then use the `hyperfirst=false` package option:

```
\usepackage[colorlinks]{hyperref}
\usepackage[hyperfirst=false]{glossaries}
```

Take care not to use non-expandable commands in PDF bookmarks. This isn't specific to the glossaries package but is a limitation of PDF bookmarks. Non-expandable commands include commands like `\gls`, `\glspl`, `\Gls` and `\Glspl`. The hyperref package provides a way of specifying alternative text for the PDF bookmarks via `\texorpdfstring`. For example:

```
\section{The \texorpdfstring{\gls{fishage}}{Fish Age}}
```

However, it's not a good idea to use commands like `\gls` in a section heading as you'll end up with the table of contents in your location list. Instead you can use

`\glsentrytext{`⟨*label*⟩`}`

This is expandable provided that the `text` key doesn't containing non-expandable code. For example, the following works:

```
\section{The \glsentrytext{fishage}}
```

and it doesn't put the table of contents in the location list.

8 Cross-References

You can add a reference to another entry in a location list using the `see={`⟨*label*⟩`}` key when you define an entry. The referenced entry must also be defined.

For example:

```
\longnewglossaryentry{devonian}{name={Devonian}}%
{%
  The geologic period spanning from the end of the
  Silurian Period to the beginning of the Carboniferous Period.

  This age was known for its remarkable variety of
  fish species.
}

\newglossaryentry{fishage}
{
  name={Fish Age},
```

```
  description={Common name for the Devonian period},
  see={devonian}
}
```

The cross-reference will appear as "*see* Devonian". You can change the "see" tag using the format `see=[`⟨*tag*⟩`]`⟨*label*⟩. For example:

```
\newglossaryentry{latinalph}
{
  name={Latin alphabet},
  description={alphabet consisting of the letters
  a, \ldots, z, A, \ldots, Z},
  see=[see also]{exlatinalph}
}
\newglossaryentry{exlatinalph}
{
   name={extended Latin alphabet},
   description={The Latin alphabet extended to include
   other letters such as ligatures or diacritics.}
}
```

If you use the `see` key in the optional argument of `\newacronym`, make sure you enclose the value in braces. For example:

```
\newacronym{ksvm}{ksvm}{kernel support vector machine}
\newacronym
 [see={[see also]{ksvm}}]
 {svm}{svm}{support vector machine}
```

9 Further Information

Further information can be found in the main glossaries user manual (glossaries-user.pdf) and there is also an article on the glossaries package on the LaTeX Community's[1] Know How section and a chapter on the glossaries package in Using LaTeX to Write a PhD Thesis.

[1] http://www.latex-community.org/