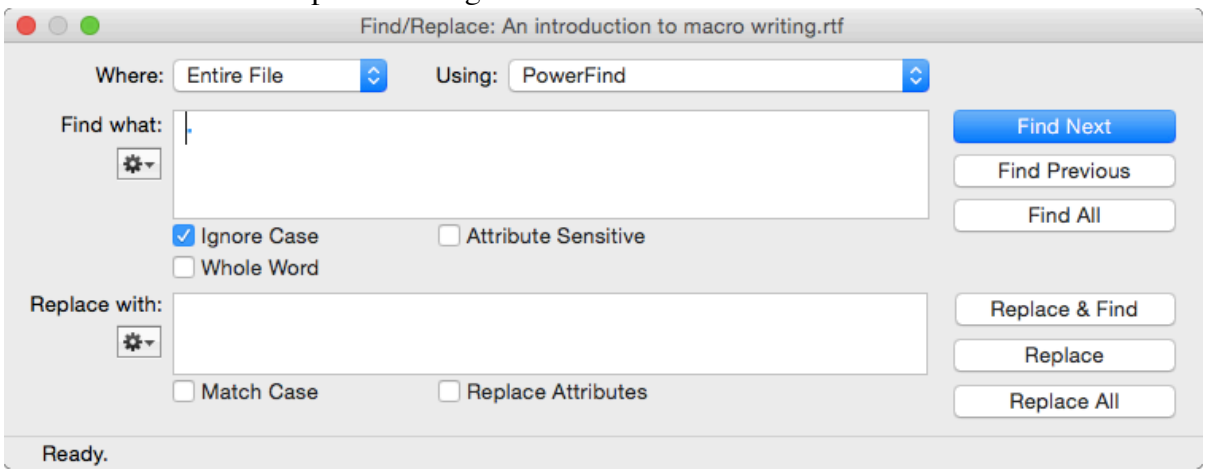
4. Find and Replace

Next we consider one more command that is so powerful it deserves its own section: Find and Replace. In fact it would probably deserve an entire book.

The Nisus "Find and Replace" dialog looks like this:



Let's consider this dialog in detail.

Central to the dialog window are the two boxes: the Find box and the Replace box. We will call whatever has been entered in the first the FIND EXPRESSION, and in the second the REPLACE EXPRESSION. What happens with these depends on the command used. The commands are to the right of the boxes. There are six options.

- Find Next
  This will find the next instance of the Find Expression in the text after the insertion point. If no instance is found before the end of the file, it *may* jump to the beginning of the file and continue searching there. The Replace Expression is ignored.
- Find Previous
  This is the same as Find Next, except the search direction is reversed.
- Find All
  This will find all instances of the Find Expression in the entire document at once. This will result in a NON-CONTIGUOUS SELECTION, if there is more than one instance of the expression in the document. The Replace Expression is not used.
- Replace & Find
  This will *first* replace the currently selected text with the Replace Expression, provided the selected text matches the Find Expression, and *then* it will move the selection to the next instance of the Find Expression, if there is one. This will only work if you have previously used 'Find (Next)' to create the selection.
- Replace
  This will replace the currently selected text with the Replace Expression, if the Find Expression is a match. The replaced text will remain selected. This will only work if you have previously used 'Find (Next)' to create the selection.
- Replace All
  This will immediately replace all instances of the Find Expression in the entire document with the Replace Expression. All replace instances will be selected as a non-contiguous selection.

Of these commands 'Find Next', 'Find All' and 'Replace All' are probably the most common and useful.

Aside from the commands there are also several OPTIONS, arranged above and below the Find and Replace boxes:

- 'Where'
  This menu contains four choices: 'Entire Document', 'In Selection', 'Here to End', and 'Here to Start'.

- 'Using'
  This menu controls the three find modes: 'Normal Find', 'Powerfind' and 'Powerfind Pro'.
- Below the Find Box are 3 checkbox options:
  'Ignore Case', 'Whole Word' and 'Attribute Sensitive'
- Below the Replace box are 2 checkbox options:
  'Match Case' and 'Replace Attribute Sensitive'

We will consider each of these options as appropriate below.
Finally there are two 'gear' menus to the left of the boxes, which we will consider in more detail. We will particularly be concerned with the following items in the Find gear menu:

- Match
- Repeat
- Special Positions
- Wild Card

And the 'Match' item in the Replace gear menu.

4.1. Basic Find operations

The basic Find operation matches the Find Expression against the text of the document character for character. Each character should match exactly as typed and in the correct order. Thus if the Find Expression is 'Nisus', it will match any instance of the exact sequence 'N-i-s-u-s' in the text. Exact match means both match for case, as well as *ignoring* any surrounding context. Consider, for instance, a Find Expression 'the'. This will of course match instances of the article *the,* but it will match all other instances of the sequence 't-h-e' as well:

The Thespians didn't wear their thermals in the theater either.

Both of these points can be modified using options. Checking the option 'Ignore Case' will mean that the expression will match both 'the', and 'The', but also 'THE', 'thE', 'tHe' and so on. Checking the option 'Whole word', meanwhile, will restrict matching to instances of the single word *the,* which must be surrounded by either space or punctuation.

4.1.1. Using Find as a macro command: Find Options

The behavior which we get from the Find dialog can be captured in a macro as follows:

```
Find 'Nisus'
```

As seen here the Find command requires a text/string argument which will act as the Find Expression for the command. A second string argument can be used to specify the options. Thus if we would like to find the next instance of the article *the,* whether capitalized or not we can write:

```
Find 'the', 'iw'
```

Here the option string contains 'i' which stands for 'Ignore Case', and 'w' which stands for 'Whole Word'. Note that the order of these options doesn't matter, so we could have used the string 'wi' to specify the same two options. Actually Nisus uses 'i' as a default option, so we could even have written 'w' alone.
What if we had wanted the macro to only match instances where the case is exactly as we have specified it? For this case we can use the option '-i', where the hyphen—symbolic for 'minus'—turns the option off. The following will match only a properly capitalized instance of the name 'Bill':

```
Find 'Bill', '-i'
```

The above will match the name 'Bill', and it won't match 'bill'. But note that it will also match the 'Bill' in 'Billings, Montana'. To make sure it matches only the name, we should use the option '-iw' or 'w-i'.
'Find' is the basic form of the command, but the macro language allows for the following variants:

```
Find Next 'Nisus'
```

```
Find Previous 'Nisus'
Find All 'Nisus'
```

As should be clear from the names of these commands, ‘Find Next’, and ‘Find Previous’ are direction specific versions of the basic ‘Find’ command, while ‘Find All’ will find all instances of the expression, and result in a non-contiguous selection.

Actually Nisus provides these commands simply as a convenience. The very same things can be achieved with the basic ‘Find’ command alone, as long as we use appropriate options:

```
Find 'Nisus'                # Same as ‘Find Next’
Find 'Nisus', 'r'           # Same as ‘Find Previous’
Find 'Nisus', 'b'           # … and this too, is ‘Find Previous’
Find 'Nisus', 'a'           # Same as ‘Find All’
```

As seen here, instead of spelling out the ‘Next’, ‘Previous’, or ‘All’ we can use options: ‘r’ mnemonic for *reverse*, ‘b’ for *backwards*, and ‘a’ for *all*. It might be noted here, that options trump the spelled out versions, so that ‘Find Next’ with an ‘r’ option, is the same as ‘Find Previous’ and with an ‘a’ it becomes the same as ‘Find All’.

Other settings of the find dialog can be handled with options as well. We can restrict the search to the current selection by using an ‘s’ option. Adding an ‘s’ option to a ‘Find’ or ‘Find Next’ command will find only the first instance—if any—in the current selection.

But more commonly we will want to find all instances of a given pattern in the selection. For this Nisus again provides a spelled out command:

```
Find All in Selection 'Nisus'
```

And as before we can get the same effect using options. The following will all work the same:

```
Find 'Nisus', 'as'
Find All 'Nisus', 's'
Find All 'Nisus', 'as'  # with a redundant ‘a’ option
```

How about a ‘Here to End’, or a ‘Here to Start’ search? For these cases there is a single option ‘W’ which is mnemonic for ‘wrap-around’, or ‘whole file’. Using this option, means that Nisus will first search in the specified direction, and then when it comes to the end of the document, it will wrap-around, and continue the search from the other end, as if the whole document were just a single loop. So the ‘W’ amounts to a whole document search. This is the default case, and there is usually no need to specify a ‘W’ option. However we can use this option to achieve a ‘Here to End’ search by turning the option off:

```
Find All 'Nisus', '-W'
Find 'Nisus', '-Wa'
```

For a ‘Here to Start’ search we need to search in the opposite direction:

```
Find All 'Nisus', '-Wr'
Find 'Nisus', '-War'
```

4.1.1. Attribute Sensitive Find

The searches we have considered to this point only consider the content of the text. But Nisus also allows search patterns to consider the formatting of the characters. This is known as ATTRIBUTE SENSITIVE FIND. A regular Find command becomes an attribute sensitive find, if we use the ‘u’ option.

```
Find '***Nisus***', 'u'
```

In this example, the expression will only match, if the text ‘Nisus’ is also both **bold** and *italic*. But actually the match will have yet more requirements. Since all my code examples in this manuscript have a Paragraph Style ‘Code’ applied to them, this style will need to be matched as well. Quite generally to get the right results with this kind of search, it’s necessary to check the applied attributes carefully, and to turn off all the attributes that are not needed. The commands ‘Format > Remove Attributes and Styles’ and ‘Remove Attribute Except Styles’ come in handy.

4.1.1. Special Characters

So far the search patterns have been limited to simple text strings. But it is quite common to want to search for more unusual characters. For instance how would one search for a paragraph marker, aka the 'carriage return' or NEWLINE character? Since trying to type one immediately creates a new line, we won't be able to get our Find Expression on a single line. However it turns out that the following *does* work correctly:

```
Find '
'
```

Since Nisus expects the first quote to be followed by a second, it will understand that the intervening newline is part of the expression. However this way of writing expressions in a macro is a bit confusing. A better solution is one that was actually mentioned earlier in the section on literals. If we use double quotes, we can use an escaped character to write the newline:

```
Find "\n"
```

The same idea can be used to include a tab character in the find expression:

```
Find "\t"
```

4.1. Powerfind and Powerfind Pro

It actually turns out that we will often want to search for things that we can't easily spell out. With the simple find expressions that we have seen so far, it is easy to find 'Chapter 1'. But in every day writing, we are much more likely to want to find the next chapter, whatever its number may be. Searching restricted to expressions that are spelled out literally, is known as NORMAL FIND. But Nisus' true strength is that it adds a more powerful and flexible mode of searching. In fact it adds two: Powerfind and Powerfind Pro.

Why two, and what's the difference?

Well it turns out that in order to add the extra power we are going to need some new ways to express what we want. This means adding many new special characters to specify these ideas. This makes it difficult to remember what those new characters are, and what they mean. To make this easier to use Nisus adds a user-friendly interface known as Powerfind. But the flip-side of this is that these user-friendly symbols require a special set of menus to enter, which makes them less convenient to type, and less portable to other systems. So Nisus adds a second method, known as Powerfind Pro, which meets those concerns.

	Powerfind	Powerfind Pro
PRO	easier to read and remember	harder to read and remember
CON	harder to enter Nisus specific	easier to enter portable to other systems

What do these two modes look like?

Powerfind	Powerfind Pro
`Find 'Chapter AnyDigit', 'e'`	`Find 'Chapter \d', 'E'`

Here I've written an expression to find the next chapter heading, no matter whether it's Chapter 2 or Chapter 9. Both expressions will work the same. The only difference is which do we find more convenient for our purposes?

The Powerfind expression uses a friendly 'bubble' that right away tips us off that something special is going on. The title of the bubble 'AnyDigit' also lets us know right away what it matches. The problem is; where do we find these bubbles? The answer is; in the Nisus Find dialog. We will have to write this kind of expression in the Nisus Find dialog; it's the only

way. But it turns out that Nisus has a handy way of getting such expressions to where you want them, called the MACROIZE feature. We'll look at that in a moment.
The Powerfind Pro expression is a little less transparent. Instead of using a bubble, it just uses text characters. That makes it easy to type, …if you know *what* to type! As we can see here, the way that Nisus knows that this isn't the usual text is, once again, by using escapes. The expression is not completely cryptic; the escape '\d' contains a 'd' which is mnemonic for 'digit'.
Finally we'll note one last important difference between the two modes: the option. Powerfind expressions will need a lowercase 'e' option, while Powerfind Pro expressions needs a capital-case 'E' option instead. If this option is missing, the expression will be treated as a Normal Find, which would mean that we would have to match an actual backslash followed by a lowercase 'd'. This is not the kind of thing you are likely to find in most English prose. But if you try it on the document you are reading here, it *will* match the expression above. So Normal Find can be handy if you are searching for macro code find expressions!

4.1.1. Wildcards

If we know the exact word or phrase we are looking for, it's easy to write a find expression for it. But often the thing we are looking for is more general. So rather than finding the specific year number '1984', we may want to find *any* year number. Or instead of finding the three word phrase 'I love you', we may want to find *any* three word phrase. For this type of search Nisus provides a number of *any-type* expressions called WILDCARDS.
The Find dialog gear menu has a Wild Card sub-menu with a long list of useful wildcards. Here we'll only consider a few of the more handy examples:

Powerfind	Powerfind Pro	matches
AnyDigit	\d *or* [[:digit:]]	1, 2, 3, … , 9, *and* 0
AnyUppercaseLetter	[[:upper:]]	A, B, C, …, Z *but also things like:* Ä, Å, Ø, É, Ç, *etc*.
AnyLowercaseLetter	[[:lower:]]	a, b, c, …, z *but also things like:* ä, å, ø, é, ç, *etc*.
AnyWordCharacter	\w	digits, letters, and the underscore '_'
AnyTextCharacter	.	digits, letters, punctuation, letters, from other alphabets, space, tab, etc. Any character that is not a newline ~~or a page break~~.

The wild card menu has some other handy things 'AnyParagraph' or 'AnyWord', which turn out to be combinations of special characters.
If we want to find a four digit year number, we could use Powerfind Pro like this:

```
Find '\d\d\d\d', 'Ew'
```

Note that I added an option 'w'. Why? Well, 'w' means 'whole word', and while this might seem strange, the computer considers '1984' to be a word. If we don't add the 'w' option here the expression will match not only four digit year numbers, but it will also match four digit sequences inside longer sequences of digits such as five digit ZIP codes, as well as any longer strings of numbers we may have in our file. It will not, however, match something like $4,000. Because in this case there is a comma interrupting the four digits. As always, the expression is very literal-minded. On the other hand, it will match both of the four digit clusters in a telephone number like this one: 090 6733 2828.

4.1.1. Repetition

Wildcards let us write some simple types of generic expressions, that are formatted in a consistent way, such as phone-numbers or ZIP codes. But often the expressions we are looking for are more open-ended. For example, our earlier chapter example. The expression we wrote works fine for 'Chapter 1' through 'Chapter 9'. But what about 'Chapter 10', 'Chapter 11', and so on? We could write:

```
Find 'Chapter \d\d', 'E'
```

But this will only work for Chapters 10–99, and *not* for Chapters 1–9, and not for higher numbered chapters either. So we would like a method that would cover all cases, no matter how many digits the number has. For this Powerfind adds REPEAT symbols. The most common are:

Powerfind	Powerfind Pro	matches
1+	+	1 or more repetitions
0+	*	0 or more repetions
0 or 1	?	1 or maybe none

These symbols need to be placed right after some other symbol, which might be a letter, or some other character, but it can also be a wildcard. Using these repeat symbols we can now improve our chapter expression like this:

```
Find 'Chapter \d+', 'E'
```

So finally this will match any numbered 'Chapter' title, from 1 to infinity!

You may be thinking; the '1+' is clear enough, but what's up with the '0+'? The '0+' really means: "We don't care if there are any or not." It's like the air around us, or for those who remember the old days, the styrofoam chips that used to come in Amazon.com shipments.

And because this repeat symbol is so flexible we should use it very carefully. Always use the '1+' or the '0 or 1', if possible, and the '0+' only if you really need it. One good rule of thumb is: Never use the '0+' with the first symbol of the expression. Always precede a character / 0+ repeat symbol combination with some fixed character or one of the Special Position symbols discussed in the next section.

Finally, what if, instead of an open-ended repetition, we would like a specific number of repetitions. For such cases it's always possible to just use repetitions of the symbols we need. Earlier we matched a four digit year number, by using the digit wildcard four times. We can match all one, or two letter words like this:

```
Find All '\w\w?', 'Ew'
```

Here the find expression matches two word characters, but the second is marked '0 or 1' so it is optional. We also add the 'w' option, otherwise it will match any sequence of 1 or 2 word characters, which would mean it would match all the word characters in the document. And since numbers are 'word' characters too, it would match all numbers as well!

For short expression like these repeating the wildcard works fine, but if we want to find larger number of repetitions—25, 38, 256, etc.—it would be quite uncomfortable to write the expression. With Powerfind Pro we can write the number of repetitions easily using braces:

```
Find '\w{7,13}', 'Ew'           # words of 7 to 13 characters
Find ' {4}', 'E'                # exactly 4 spaces
Find '\d{,3}', 'Ew'             # numbers with at most 3 digits
```

Actually the last expression will also match 'numbers' with *no* digits, i.e., nothing!

To make Powerfind expressions like this, Nisus has a special 'bubble creation' dialog in the Repeat sub-menu of the gear menu.

```
Find ' AnyWordCharacter _7-13 Times', 'ew'
Find ' Space 4 Times', 'e'
Find ' AnyDigit _0-3 Times', 'ew'
```

4.1.1. Special Positions

Earlier we wrote a find expression to find chapter headings. But what if our chapters are not conveniently labelled with the word 'Chapter'? Maybe our chapter headings simply begin with a number followed by a space and some text. We could write this:

```
Find '\d+ .+', 'E'
Find ' AnyDigit 1+ AnyTextCharacter 1+ ', 'e'
```

The problem is that while this will match the chapter heading

1 Introduction

it will also find the following

Marilyn bought 300 pencils and sold half of them.

The difference is that our headings will have the number at the beginning of a paragraph. In order to allow us to make such distinctions, there are a number of SPECIAL POSITION symbols.

Powerfind	Powerfind Pro
Start of Paragraph	^
End of Paragraph	$
Start of Word	\<
End of Word	\>

Using these we can improve our earlier find expression like this:

```
Find '^\d+ .+', 'E'
Find ' Start of Paragraph AnyDigit 1+ AnyTextCharacter 1+ ', 'e'
```

Adding the start of paragraph symbol will ensure that only paragraphs which start with a digit will be matched.

One of the most common find tasks asks us to match a paragraph. Since a paragraph consists of some amount of text, followed by a return character, we can make a find expression like this:

```
Find '.+\n', 'E'
Find ' AnyTextCharacter 1+ Return ', 'e'
```

And while these expressions will usually work, they may sometimes disappoint us, by not selecting the whole paragraph. If we add a 'start of paragraph' symbol at the beginning, we will be sure that only whole paragraphs are matched.

```
Find '^.+\n', 'E'
Find ' Start of Paragraph AnyTextCharacter 1+ Return ', 'e'
```

This use of the start symbol is even more important when we want to use the '0+' repeat symbol. Let's consider the case where we want to find all paragraphs that contain the phrase 'Nisus Writer', that is, we want to match the phrase as well as any text before or after it. Since the phrase might be at the very beginning of the paragraph or at the very end, we need to allow for the possibility that there is no text before it, as well as no text after it. The best way to write this is like this:

```
Find '^.*Nisus Writer.*\n', 'Ea'
```

We can spell this out as follows:

^	start of paragraph
.*	any amount of text (possibly none)
Nisus Writer	our desired search phrase
.*	more text (again possibly none)
\n	return character

Note that in a case like this it will be imperative to use the start of paragraph delimiter. Without this, the find expression will be extremely slow to complete, or it may even fail to complete at all and require that Nisus be shut down. The exact reason for this is rather technical, but as a general rule of thumb it will be good to remember to avoid beginning find

expressions with the expression '.*' or 'AnyTextCharacter 0+' or for that matter any expression involving the '0+' repeat symbol. Always precede such expressions with a fixed character/phrase or a special position delimiter.

4.1.1. Matching

Now that we know how to find our chapter headings, imagine we want to rearrange them; that is we would like to transform

1 Introduction

to

Chapter 1: Introduction

We can use the Find expression

```
Find '^\d+ .+', 'E'
```

to find the chapter headings, but when we want to write the replace expression, we face a problem. How do we tell Nisus to use the number and title we found, as part of the replacement?

To solve this type of problem Powerfind has matching. As part of the find operation we can ask Powerfind to remember specific found bits and then refer back to them, much the same way we do with variables. To identify—and refer back to—these bits the match menu provides the following expressions:

Powerfind	Powerfind Pro	purpose
Capture(…)	(…)	creates a capture
Captured1	\1	refers back to the first capture
Captured2	\2	refers back to the second capture
…	…	etc.

We can use them like this:

```
Find and Replace '^(\d+) (.+)', 'Chapter \1: \2', 'E'
```

or using Powerfind:

```
Find and Replace " Start of Paragraph Capture( AnyDigit 1+ )
Capture( AnyTextCharacter 1+ )", "Chapter Captured1 :
Captured2", 'e'
```

In this expression we have used the 'capture' parentheses around both the digit expression and the expression that catches the text of the chapter title. These captured expressions are then numbered from left to right and we can refer back to them using the special symbols '\1' for the first one and '\2' for the second. More such expressions can be added as needed.

Just to be clear; the numbering only refers to the finding order. Once the find expression has matched the different bits, we can use them as part of the replace expression in any order, and as often as we would like. This means that we can use this method to duplicate things, or to reorder them:

```
Find and Replace '^(\w+)', '\1 \1', 'E'
Find and Replace '(.)(.)', '\2\1', 'E'
```

The first of these two expressions will find the first word of the next paragraph, and repeat it separated by a space. Admittedly not a very useful thing to do. The second expression will 'toggle' the next two characters. This is slightly more useful.

4.1.1. Custom-made Wildcards

Earlier we saw that Powerfind has a number of wildcards to match things like digits, letters, and word charcters. But sometimes we would like our wildcards to be much more specific. For instance '\w+' will match any word, but maybe we would like to match only one of a (small) collection of specfic words.

Consider the case of matching only forms of the verb ‘be’, which has eight different forms: ‘am’, ‘are’, ‘is’, ‘was’, ‘were’, ‘be’, ‘being’, and ‘been’. This can be accomplished using the Powerfind OR symbol. We can string the different words we are looking for together, separated by ‘or’ symbol. If we enclose the list of selections within capture parentheses, we can later refer back to the specific form we have found:

```
Find '(am|are|is|was|were|be|being|been)', 'Ew'
Find ' Capture( am Or are Or is Or was Or were Or be Or
       being Or been ) ', 'ew'
```

The ‘or’ symbol is not restricted to combining full words and expressions. It can also be used to combine parts of an expression. For this to work correctly capture brackets will need to be used. For example if we want to match any of the four possible forms of the verb ‘talk’, we can write the expression like this:

```
Find 'talk(s|ing|ed)?', 'Ew'
Find 'talk Capture( s Or ing Or ed ) 0 or 1 ', 'ew'
```

The four forms of the verb ‘talk’ are ‘talk’, ‘talks’, ‘talking’ and ‘talked’. Of course we could combine all four of these forms using the ‘or’ symbol, but since ‘talk’ is shared by all four forms there is no need to repeat this. Instead we can limit the ‘or’ to apply only to the three endings ‘-s’, ‘-ing’ and ‘-ed’. To match the plain form without any ending we tag the capture group with the ‘?’ (‘0 or 1’) repeat symbol.

Another type of custom wildcard is the CHARACTER SET. The built-in wildcard ‘.’ matches any character, while the ‘\w’ matches only word characters. But what if we want to restrict ourselves to a more limited group of characters? The character set brackets make this possible. For example we can match all vowels and only the vowels like this:

```
Find '[aeiou]', 'E'
Find ' CharacterInSet[ aeiou ] ', 'e'
```

In Powerfind Pro we can create a character set simply by enclosing the desired options in square brackets. The same can be achieved in Powerfind using the ‘CharacterInSet’ brackets found in the ‘Match’ menu. The entire group will match only a single character, as long as the matched character is any one of the items inside the brackets.

Nisus Writer uses these character set brackets to pre-combine several wildcards: the wildcards ‘Any Space’, ‘Any Double Quote’ and ‘Any Single Quote’ are all formed in this way, although this will only be apparent, if you look at the Powerfind Pro version.

Sometimes it is easier to say which charcters that you don’t want to match. For this eventuality Powerfind provides the ‘CharacterNotInSet’ brackets. The Powerfind Pro equivalent is to add a caret (‘^’) as the first symbol inside the character set brackets.

```
Find '[^\t]+', 'E'
Find ' CharacterNotInSet[     ] 1+ ', 'e'
```

This expression will match a string of characters where none are tab characters.

Summary:

Powerfind	Powerfind Pro	matches
… Or …	… | …	
CharacterInSet[…]	[…]	
CharacterNotInSet[…]	[^…]	

4.1.1. Custom-made Position Symbols

Matching text that’s surrounded by smart quotes.

```
Find '(?<=“).+?(?=”)', 'E'
```

Powerfind	Powerfind Pro	matches

FollowedBy(…)	(?= …)	
PrecededBy(…)	(?<= …)	

4.1.1. Macroize

So about that Macroize feature: If we want to create a Powerfind expression—or if we can't remember the proper escape to write a Powerfind Pro expression—we can use the Nisus Find dialog. Place the cursor in the find box and select the appropriate bubbles from the sub-menus in the gear menu. Do the same for the replace expression, if required. Make sure to choose and/or check all the relevant options. Once you're satisfied with the expression, select 'Macroize…' from either the Find or the Replace gear menu. (It does not matter which, as both work the same.) This will bring up a dialog that allows you to choose the Find command from a pop-up menu. There are six choices: Find Next, Find Previous, Find All, Replace Next, Replace Previous, Replace All. Finally there are two ways to complete the procedure by saving the whole expression either to a new macro, or to the clipboard. The latter case is useful when we want to add the expression to an already existing macro.

4.1. Some useful examples

Find/Delete identical repeated paragraphs

```
Find and Replace '^(.+\n)\1+', '\1', 'Ea'
```

Swap tab separated (specific) columns:
1st and 2nd:

```
Find and Replace '^([^\t]*)\t([^\t]*)', '\2\t\1', 'Ea'
```

2nd and 3rd:

```
Find and Replace '^([^\t]*\t)([^\t]*)\t([^\t]*)', '\1\3\t\2', 'Ea'
```