TEXcount
Perl script for counting words in LATEX documents
Version 3.1

Abstract

TEXcount is a Perl script for counting words in LATEX documents. It recognises most of the common macros, and has rules for which parameters to count and not to count; the main text is counted separately from the words in headers and in captions of figures and tables. Finally, it produces a colour coded version of the parsed document, either as a text document or as HTML to be viewed in a browser, indicating which parts of the document have been included in the count.

Contents

1 What TEXcount does

TEXcount is a Perl script made for counting the words in a LATEX document. Since LATEX documents are formated using lots of macro instructions and often contain both mathematical formulae and floating tables and figures, this is no trivial task.

Simple solutions to counting the words consists of detexing the documents, which often merely consisty of ignoring the TEX and LATEX instructions. This is a bad solution since it will usually result in over-estimating the number of words as mathematical formulae, citations, labels and references are counted.

A perfect solution, if such exists, needs to take into account how LATEX interprets each macro instruction. The simplest approach to taking this into account consisty of making the count based on the typeset document, but this too tends to over-estimate the word count as mathematical formulae, table contents and even page numbers may get counted.

A simple but robust approach, which is the one I have taken with TEXcount, is to parse the LATEX document using simple rules for how to interpret the different TEX and LATEX instructions. Rules for most of the common macro instructions are included in TEXcount, and it is possible to specify new rules in the TEX document.

The primary focus of TEXcount is to:

- provide an accurate count of the number of words in LATEXdocuments;
- exclude or count separately document elements which are not part of the main text such as figure captions;
- enable the user to, with relative ease, check how TEXcount has parsed the document and which elements have been counted and which have not.

The last point on this list is one of the most important. Having an accurate word count is of little value unless you know that it is accurate; conversly, trusting an inaccurate word count can be potentially harmful, e.g. if you are submitting a paper or a report which has a strict word limit.

TEXcount handles complete LATEX documents, i.e. that start with \documentclass and has the text between \begin{document} and \end{document}, as well as partial documents made to be included in another LATEX document. However, in either case, it requires that all groups are closed: {...} and \begin...\end.

Automatic parsing of included documents is possible, but is by default turned off. There are two options for turning this on: -inc and -merge. Turning it on using -merge will merge the included files into the main document. By using -inc, however, the included files are parsed separately rather than include the text into the appropriate location: this will perform a separate word count of the included document which is then later included in the total sum.

Since TEXcount relies on a relatively simple rules for handling the different macros and only performs limited checks on the validity of the LATEX document, it is your responsibility to make sure the document actually typesets in LATEX before running it through TEXcount. Also, TEXcount relies on handling macros taking parameters enclosed with { and }, and on ignoring options enclosed by [and]: macros with significantly different syntax such as \vskip cannot be handled. There are also limitations on what may be contained in macro options enclosed in [], although this restriction may be relaxed by specifying the command line option -relaxed.

1.1 What TEXcount counts

Basically, TEXcount has seven different counts plus an additional file count for use with total counts over a set of files. These and their indices (numbers used to identify them) are:

0. Number of files: When multiple files are included, this is counted.

1. Text words: Words that occur in the main text.

2. Header words: Words that occur in headers, e.g. \title and \section.

3. Caption words: Words that occur in figure and table captions.

4. **Header count:** This counts the number of headers, i.e. each \section counts as 1.

5. **Figure/float count:** This counts the number of floats and figures, e.g. table and figure environments.

6. **Inline formulae:** This counts the number of inline formulae, i.e. $...$.

7. **Displayed formulae:** This counts the number of displayed formulae, e.g. \[...\] or equation environments.

These are stored in an array and sometimes referenced by their index: e.g. in the option -sum= which takes parameter values corresponding to counts 1 to 7. In other contexts, however, like in the -tempate= or when incrementing specific counters through the %TC:macrocount instruction, the counters may be referred to by keywords rather than the indices 0 to 7.

There is also support for adding additional counters using the TEXcount instruction %TC:newcounter. These will then be added to the end of the list of counters and should preferably be referred to by name, not index.

1.1.1 What TEXcount counts as a word

The primary role is to count the words. It is not entirely clear what should be considered words, so I have had to make some decisions. A sequence of letters is certainly a word. TEXcount also counts acronyms like *e.g.*, dashed words like *over-all*, and *it's* as one word. It also counts numbers as words unless they are placed in a math group. If TEXcount breaks words that contain special characters, you may try the option -relaxed which extends the range of characters allowed as part of words.

Alternatively, TEXcount may be asked to count the number of letters/characters (not including spaces). It may also be set to count Chinese or Japanese characters.

Mathematical formulae are not counted as words: it would be difficult to define a sensible rule for this. Instead, TEXcount counts the number of inline formulae and displayed formulae separately. You may then decide on how to combine these counts with the word counts, e.g. using the -sum option.

Text in headers (\title, \section, etc.) are counted separately: TEXcount counts the number of headers as well as the number of words in headers. It may also provide subcounts for each of these by specifying the -sub option.

Floating environments (or potentially floating environments) such as tables and figures are not counted as text, even if the cells of a table may contaiint text. However, if they have captions, these will be counted separately much like headers were. Footnotes are included in this count. By default, environments do not modify the parsing state: i.e. environments within the text are counted as text, etc. Rules for the most common environments, at least those that require non-default treatment, should be predefined, but you may have to add more rules if you use environments defined in packages or by yourself. If you wish to be warned against any environments names you use that lack a defined rule, set the option -strict.

Some macros are words by themselves: e.g. \LaTeX. These are counted as words provided the macro word rule has been defined for them, but you cannot expect TEXcount to count something like \LaTeX-word or {\TeX}count as one word although the above explanation inicates that it should: TEXcount will in both cases evaluate the macro and the following text separately and thus count them as separate entities. Since TEXcount recognises \LaTeX and \TeX as single words, each of the two examples would end up being counted as two words.

1.2 What TEXcount does not do

While an ideal solution should be able to expand the macro instructions, thus being able to handle new macros, that would at it's worst require reimplementing much of TEX, something that is clearly unrealistic. Instead, I have opted for a simpler solution: to define rules stating which paramters to count and which to ignore and allowing for such rules to be added easily. Thus, TEXcount cannot handle macros that may take a variable number of parameters. Nor can it handle macros that takes parameters on forms other than {parameter}. However, support has now been added for macro options on the form [...] to be parsed.

In general, while TEXcount does the parsing in some detail, it does not do it exacly as TEX does it. In some respects there may therefore be critical differences: e.g. while TEX reads one character at a time,

TEXcount reads one word at a time, so while LATEX would interpret \cite me as \cite{m}e, TEXcount would interpret it like \cite{me}.

Another issue is that, since TEXcount does not know how to expand macros, it cannot handle macros like \maketitle that add text to the document. With respect to \maketitle, I have instead set the rule for \title{title text} to count this as a header although it does not itself produce any text.

1.3 Problems to be aware of

In most large documents, there will be cases where TEXcount does not give an exact count. Reasons may be macros TEXcount does not recognise, words that TEXcount split in two (or more) because of special characters not recognised as letters, or options and parameters not counted which actually produce text. Some problems may also arise because it is not always clear what should be counted and TEXcount implements one particular choice: counting numbers as letters/words, not counting formulae as words, not to count tables as text, etc. However, hopefully these should either consist of individual, infrequent errors which should have limited effect on the total count, or entire regions that are included or excluded for which the user may change the parsing rule to produce the desired count.

There are, however, problems that may arise which are more fundamental and result in counts which are simply wrong rather than just inaccurate, or even make TEXcount fail entirely.

If TEXcount fails to detect environment endings properly, either closing { or \end, it may end up ignoring major parts of the document. This should normally produce errors of some kind, although there may be cases when no errors are produced. However, by looking at the verbose output, it will be very clear that entire parts of the document has been excluded. Such problems may be cause by macros that allow unmatched group delimiters, and some effort has been made to minimise the risk of this at the cost of risking other but less critical errors: e.g. there are limits to what is permitted as macro options in order to ensure that a single unmatched [does not cause large parts of the document to be interpreted as a big option.

For users of languages containing letters other than the Latin letters A to Z, there is a risk that TEXcount may have difficulty identifying words correctly. The script relies on Perl to recognise words as sequence of letters, and must therefore know which characters are considered to be letters. Words containing letters not recognised by TEXcount will tend to be split into two or more words, which can dramatically inflate the word count. The first step is to ensure that the file is read using the correct encoding: I generally suggest using the UTF-8 Unicode encoding, and from version 2.3. this is the default encoding used by TEXcount, although other encodings may also be used. Unicode has good annotation of which characters are letters, and starting with version 2.3, TEXcount uses Unicode internally to represent the text.

While non-Latin letters like å and ä should be recognised as letters, TEX/LATEXcodes using macros or special characters, such as \aa and \"a, are not immediately understood as letters. I have added patterns aimed at recognising these as well, but depending on the code you are writing, these patterns may either not be flexible enough to recognise all letter codes, or may be too flexible and recognise things it should not. I have added a relaxed mode (-relaxed) and a more restricted mode (-restricted) in which these patterns are more general or more constrained, but you should check how this performs on you actual texts by viewing the verbose output.

2 Syntax and options

2.1 Running TEXcount

The command to run TEXcount may vary slightly depending on the operating system and the local settings. You may also wish to rename it or define an alias.

Under Windows, running texcount.pl from the command line suffices if texcount.pl is in the path and pl-files are defined to run as Perl scripts.

Under Linux/Unix, it should be sufficient to run texcount.pl provided it is in the PATH and has been made executable (chmod u+x texcount.pl). The first line of the file contains the line #!/usr/bin/env perl which

should find the correct location for perl (provided the program /usr/bin/env is available). If not, run which perl to locate Perl and replace the first line of the script with #!*path*.

Alternatively, if the above methods do not work, you may have to run TEXcount exclicitly in Perl by executing perl texcount.pl. You then need to have the perl executable in the path or give the explicit path.

2.2 TEXcount command-line options

For simplicity, I will simply write texcount.pl in this manual for the code to execute the script. The syntax then becomes

texcount.pl *[options] [files]*

where the options may be amongst the following:

-v Verbose (same as -v3).

-v0 No details (default).

-v1 Prints counted text, marks formulae.

-v2 Also prints ignored text.

-v3 Also includes comments and options.

-v4 Same as -v3 -showstate.

-v=..., -v[[0-4]]... Allows detailed control of which elements are included in the verbose output. The provided values is a list of styles or style categories separated by + or - to indicate if they should be added or removed from the list of included styles. Style categories start with capital letter and include Words, Macros, Options; the individual styles are in all lower case and include word, hword, option, ignore.

-showstate Show internal states (with verbose).

-brief Only prints a one line summary of the counts for each file.

-q, -quiet Quiet mode, does not print error messages. Use is discouraged, but it may be useful when piping the output into another application.

-strict Prints a warning of begin-end groups for which no specific rule is defined.

-total Only give total sum, no per file sums.

-1 Same as specifying -brief and -total, and ensures there will only be one line of output. If used with -sum, the output will only be the total number.

-0 Same as -1, i.e. -brief and -total, but does not put a line shift at the end. This may be useful when the one line output is to be used by another application, e.g. Emacs, for which the line shift would otherwise need to be stripped away.

-template="..." Specify an output template which is used to generate the summary output for each file and for the total count. Codes {*label*} is used to include values, where *label* is one of 0 to 7 (for the counts), SUM, ERROR or TITLE (first character of label is sufficient). Conditional inclusion is done using {*label*?*text*?*label*} or {*label*?*if non-zero*|*if zero*?*label*}. If the count contains at least two subcounts, use {SUB?*template*?SUB} with a separate template for the subcounts, or {SUB?*prefix*|*template*|*suffix*?SUB}.

-sub[=...], -subcount[=...] Generate subcounts. Valid option values are none, part, chapter, section and subsection (default), indicating at which level subcounts are generated. (On by default.)

-nosub Do not generate subcounts.

-sum[=n,n,...] Produces total sum, default being all words and formulae, but customisable to any weighted sum of the seven counts (list of weights for text words, header words, caption words, headers, floats, inlined formulae, displayed formulae).

-nosum Do not generate total sum. (Default choice.)

-col Use ANSI colour codes in verbose output. This requires ANSI colours which is used on Linux, but may not be available under Windows. On by default on non-Windows systems.

-nc, -nocol No colours (colours require ANSI). Default under Windows.

-nosep, -noseparator No separating character/string added after each word in the verbose output (default).

-sep=, -separator= Separating character or string to be added after each word in the verbose output.

-relaxed Relaxes the rules for matching words and macro options.

-restricted Restricts the rules for matching words and macro options.

- Read LaTeX code from STDIN.

-inc Parse included files (as separate files).

-merge Merge included files into document (in place).

-noinc Do not parse or merge in included files (default).

-incbib Include bibliography in count, include bbl file if needed.

-nobib Do not include bibliography in count (default).

-incpackage= Include rules for a given package.

-dir[=...] Specify working directory which will serve as root for all include files. The default (-dir=.) is to use the current directory, i.e. from which TeXcount is executed: the path can be absolute or relative to the current directory. Use -dir to use the path of the main LaTeX document as working directory.

-auxdir[=...] Specify the directory of the auxiliary files, e.g. the bibliography (.bbl) file. The default setting (-auxdir only) indicates that auxiliary files are in the working directory (from the -dir or -dir= option). If -auxdir= is used to provide a path and -dir= is used to specify the working directory, the path to the auxiliary directory is take to be relative to the current folder (from which TeXcount is executed); if used with -dir, the working directory is taken from the path of the parsed file, and the auxiliary directory is taken to be relative to this (unless an absolute path is provided).

-enc=, -encoding= Specify encoding to use in input (and text output).

-utf8, -unicode Use UTF-8 (Unicode) encoding. Same as -encoding=utf8.

-alpha=, -alphabets= List of Unicode character groups (or digit, alphabetic) permitted as letters. Names are separated by , or +. If list starts with +, the alphabets will be added to those already included. The default is Digit+alphabetic.

-logo=, -logograms= List of Unicode character groups interpreted as whole word characters, e.g. Han for Chinese characters. Names are separated by , or +. If list starts with +, the alphabets will be added to those already included. By default, this is set to include Ideographic, Katakana, Hiragana, Thai and Lao.

-ch, -chinese, -zhongwen Turn on Chinese mode in which Chinese characters are counted. I recommend using UTF-8, although TEXcount will also test other encodings (GB2312, Big5, Hz) if UTF-8 fails, and other encodings may be specified by -encoding=.

-jp, -japanese Turn on Japanese mode in which Japanese characters (Kanji and Kana) are counted. I recommend using UTF-8, although TEXcount will also test other encodings (e.g. EUC-JP) if UTF-8 fails, and other encodings may be specified by -encoding=.

-kr, -korean Turn on Korean mode in which Korean characters (Hangul and Han) are counted. I recommend using UTF-8, although TEXcount will also test other encodings (e.g. EUC-KR) if UTF-8 fails, and other encodings may be specified by -encoding=.

-kr-words, -korean-words Korean mode in which Hangul words are counted (i.e. as words separated by spaces) rather than characters. Han characters are still counted as characters. See also -korean.

-chinese-only, ..., -korean-words-only As options -chinese, ..., -korean-words, but also excludes other alphabets (e.g. letter-based words) and logographic characters.

-count-all, -all Allow all alphabets, digits, and logograms as letters, whether counting words or characters. This is the default setting.

-char, -letter Count letters instead of words. This count does not include spaces.

-char-only, -letter-only Count letters instead of words, but excludes logograms (like chinese characters). This count does not include spaces.

-all-nonspace-char, -all-nonspace-characters Counts characters instead of words, including letters, logograms, and punctuation, but not including spaces.

-out= Send output to file. Takes file name as value.

-out-stderr Send output to STDERR instead of STDOUT.

-html Output in HTML format.

-htmlcore Only HTML body contents.

-htmlfile= File containing a template HTML document with <!– TeXcount –> included somewhere to indicate the location where the TeXcount output from the parsing should be inserted.

-tex Encode TEX special characters for output into TEX code.

-css= Reference to CSS to be included in the HTML output instead of including the style definition directly in the output.

-cssfile=, -css=file: File containing style definitions to be included into the HTML output instead of the default styles.

-freq[=#] Count individual word frequencies. Optionally, give minimal frequency required to be included in output.

-stat Produce statistics on language usage, i.e. based on the alphabets and logograms included.

-macrostat, -macrofreq Produce statistics on package, environment and macro usage.

-codes Display an overview of the colour codes. Can be used as a separate option to only display the colour codes, or together with files to parse.

-nocodes Do not display overview of colour codes.

-opt=, -optionfile= Reads options (command line parameters) from a specified text file. Should use one option per line. May also include TC options in the same format as specified in LaTeX documents, but prefixed by % rather than %TC:. Blank lines and lines starting with # are ignored; lines starting with \ are considered to be continuations of the previous line.

-split, -nosplit The -split option, which is on by default, speeds up handling of large files by splitting the file into paragraphs. To turn it off, use the -nosplit option.

-showver, -nover Include version number in output with -showver; use -nover not to show it (default).

-h, -?, –help, /? Help.

-h=, -?=, –help=, /?= Help on particular macro or group name: gives the parsing rule for that macro or group if defined. If the the macro or environment is package specific, use -h=*package*:*name*; replace *package* with class%*name* if it is specific to a document class.

-help-options, -h-opt Lists all TeXcount options and help on them.

–help-option=, -h-opt= Lists all TeXcount options containing the provided string: e.g. -h-opt=inc lists all options containing inc, while -h-opt=-v lists all options starting with v.

-help-style, -h-style Lists all styles and style categories, i.e. those permitted used in -v=styles-list.

-help-style=, -h-style= Gives description of style or style category.

-ver, –version Print version number.

-lic, –license License information.

If more than one file is given, TeXcount will perform the count on each of them printing the results for each, then print the total sum at the end. Note that files are parsed one by one in order of appearance and counts made per file; only afterwards are the totals computed.

2.2.1 Option handling alternatives and modifications

In some cases, eg when running from within a script, the option syntax may cause problems. Two minor modifications have been added.

For set options on the form -[options]=[value], you may use the alternative syntax -[option]:[value] to avoid any problems caused by the equal character.

TeXcount will ignore any option starting with @- which can either be used to temporarily exclude an option in editing an existing call to TeXcount, or to use the @-[option] to pass options to scripts without having to remove these before passing the arguments on to TeXcount.

2.3 File encoding

If your TeX/LaTeX document consists entirely of ASCII characters, there should be no problems with file encoding. However, if it contains non-ASCII characters, e.g. non-Latin letters such as ø, there are different ways in which these may be encoded in the files.

The main encoding supported by TeXcount is UTF-8 (Unicode), and this is used to represent text internally in TeXcount. In older versions of TeXcount, Latin-1 (ISO-8859-1) was the default encoding, but this may cause problems when using non-Latin characters.[1] Both of these are compatible with ASCII: i.e. both are extensions of ASCII, so ASCII characters will be treated correctly by both encodings, but non-ASCII characters will be treated differently.

From version 2.3 of TeXcount, it is possible to specify other encodings using the -encoding= option. If no encoding is specified, TeXcount will guess which encoding is used. By default, this guessing is limited

[1] In Perl, which TeXcount is written in, Latin-1 is the default. However, starting with version 2.3, TeXcount has switched to using UTF-8 (Unicode) internally and will convert text to Unicode before processing: in older version, internal representation was UTF-8 or Latin-1 depending on the options used.

to ASCII, UTF-8 and Latin-1. If other encodings are used, the automatic guessing is likely to pick Latin-1 since most files would result in valid Latin-1 code. If the -chinese or -japanese option is set, it will guess at other encodings, but still with UTF-8 as the first choice.

I generally recommend using UTF-8 Unicode: this is increasingly being the new standard. Basically, Unicode contains the characters needed for all existing languages, enumerated from 0 and upwards (beyond 100000), which resolves to problem of requiring different character sets. Since there are more than 256 characters in Unicode, Unicode cannot be represented using one byte per character: UTF-8 is a way to encode the Unicode characters into a list of bytes so that ASCII characters (no. 0–127) are represented by one byte (same as in ASCII), while non-ASCII characters are represented using two or more bytes. Unicode may also be encoded using two bytes to represent each of Unicode characters 0–65535, which covers most of practical use, but this is less commonly used as a file format: it is, however, common for internal representation of strings in memory, as done by e.g. Java, so Perl is the odd one out in using UTF-8 for internal string representation.

If an encoding is specified using the -encoding= option, the input will be decoded from the specified encoding into UTF-8. If HTML output is specified, the output will be UTF-8. This ensures that all HTML produced is UTF-8, which is also the encoding specified in the HTML header. If text output is used, the specified encoding is used for the output. E.g. if you specify -encoding=latin1, TEXcount will assume that all files are encoded in Latin-1, and will also produce the detailed output using Latin-1. For piping, i.e. option -, this is useful as it ensures the output has the same encoding as the input.

For convenience, if no encoding is specified, TEXcount will try to guess which encoding is the appropriate one. This is done simply by checking a specified list of encodings one by one until one is found that fits the text. The default is to check ASCII, then UTF-8, and finally Latin-1. If none fits, TEXcount should try to decode the ASCII part of the text replacing non-ASCII characters with a wildcard character, although there may be cases when the decoding exits upon hitting an error. If Chinese or Japanese languages are specified, UTF-8 is tried first, then other encodings are checked depending on the language.

Note that if no encoding is specified and TEXcount left to guess the appropriate encoding, all output will be UTF-8. Thus, letting TEXcount guess the encoding may not be suitable when using TEXcount in a pipe since the UTF-8 output may not be compatible with the encoding of the input. If multiple files are parsed, TEXcount will guess the encoding separately for each file even if they are included (-inc or -merge) in a file with an identified encoding, and may thus end up selecting different encodings for different files.

2.4 Language scripts, alphabets and character sets

In additional to the traditional Latin letters, A-Z, a number of letters are recognised by Unicode as part of the extension of the Latin letters. Some languages, however, use entirely different character sets.

By default, TEXcount has been set up to recognise all alphabets. However, there is a distinction between alphabets like the Latin, Greek, Cyrillic, etc. in which words consists of multiple letters, and languages like Chinese in which each character should be counted as a word. For simplicity, we refer to these as *alphabetic* characters and *logograms*.[2] The options -alphabets= and -logograms= (or -alpha= and -logo= for short) allows you to specify which characters to use as either alphabetic letters or whole word characters. These take values that consist of Unicode properties separated by , or +. The default setting corresponds to

```
-alphabets=Digit,alphabetic
```

in which alphabetic is defined by TEXcount as the Unicode Alphabetic class minus logographic script classes, and

```
-logograms=Ideographic,Hiragana,Katakana,Thai,Lao
```

which should cover Chinese characters (Han) as well as the Japanese characters (Han for the kanji, Hiragana and Katakana for the kana). Both options remove previous script settings, unless the list is prefixed by + in which case the scripts are added: e.g. -logograms=+cjkpunctuation will add the CJK punctuation characters (defined by TEXcount) to the set of counted characters.

[2] Actually, these names are not completely accurate. A logogram is a script which represents a word or 'meaningful unit', but e.g. the Japanese kana and Korean hangul are counted as words although they represent sound or syllables rather than meanings.

Applicable Unicode properties/scripts include Digit, Latin, Greek, Cyrillic, Hebrew, Arabic, Han, Katakana, Hiragana, and more.[3]

In addition to the Unicode properties, TEXcount has added a few additional character groups. The properties alphabetic, digit and alphanumeric are more restrictive than their Unicode name-sakes: alphabetic excludes the default logographic character sets, and digit consists only of 0–9 unlike Unicode Digit which includes numerals from other scripts. There is also cjkpunctuation which is intended to identify Chinese/-Japanese/Korean punctuation.

Note that the Unicode properties are case sensitive. The native Unicode properties start with capital letters, whereas the properties defined by TEXcount are all lower case. Invalid properties will be ignored.

The options -chinese and -japanese still exist and simply restrict the logographic character sets. In addition, -chinese-only and -japanese-only will exclude alphabetic words from the counting, equivalent to -alphabets= with no script properties given. In addition, these options will change the lists of file encodings TEXcount will try if no encoding is given.

The option -stat has been added to produce overall word counts per script type. This uses the character classes specified in the -alphabets= and -logograms= options, so the default will be able to count which words are purely alphabetic and which contain numbers (or a combination of both), but will not distinguish between e.g. Latin and Greek. To do that, you would have to specify the script classes: e.g.

```
-alphabets=digit,Latin,Greek,Cyrillic
```

will count words containing the numbers 0–9, Latin letters (including the extended Latin character set), Greek letters and Cyrillic letters. Words may contain any combination of these: TEXcount does not require that a word consist of only one type of script. Also, note that if digit had not been included, numbers would not be allowed to be part of or counted as words. The output statistics will then give the number of words containing each of these script classes (or combination).

2.5 Parsing details

By selecting one of the -v options, you can choose how much detail is printed. This is useful for checking what TEXcount counts. Alternatives -v0 to -v4 control the amount of detail, with -v equal to -v3. The option -showstate shows the internal state and is for debugging purposes only: -v4 switches this on.

The output is colour coded with counted text coloured blue, other colours for other contexts. The colour coding is made using ANSI colour codes. These should work when printed directly to Linux xterm window, but need not work if piped through more or less: with less you need to use the option -r for the colours to be shown correcly.

Under Windows or other operating systems, regular ANSI colour don't work, but there is a fix in place which adapts it for Windows, although this may not function exactly as desired.

In general, however, I recommend using HTML output which can be viewed in a browser: in particular if the text output does not produce suitable colour coding.

To print the details encoded as HTML document, use the option -html. Alternatively, -htmlcore only outputs the HTML body. I suggest using the options -html -v to get full detail, save this to a HTML file, e.g. using

```
texcount.pl -html -v -sum files > details.html
```

where -sum computes the total count of words and formulae (or -sum=1,1,1 to only count words) and adds the cumulative count at the end of each line of the parsing details, and -sub is on by default which produces subcounts per section.

2.5.1 Control of details in verbose output

The verbosity option, -v=*styles-list* or -v[0-4]*styles-list*, can be used to select exactly which elements to include or exclude from the verbose output. The styles list consists of a list of styles or style categories with +*style* or -*style* used to indicate if they should be added or removed. If the first style in the list is one of the

[3] A more complete overview is available at Wikipedia: `http://en.wikipedia.org/wiki/Script_(Unicode)`.

categories 0 to 4, the = can be dropped. The option -help-style returns an overview of the available styles and style categories, while -help-style=*style* may be used to get a description of a particular style or style category.

Each token in the verbose output has a defined style: e.g. word, hword (header word), ignore, option, etc. If the style is included in the styles list, it will be printed in the verbose output; if not included in the styles list, it will not be printed. Thus, by setting which styles are included in the styles list, you can specify in detail which tokens are included in the verbose output. The included styles correspond to the list of colour codes listed at the start of the output when -codes is set.

The style categories, which include 0 to 4, are groups of related styles: e.g. Words, Macros, Options, etc. Note that apart from 0 to 4, the style categories have capital initials, while the styles themselves are all lower case.

For example, if you only want to output words (including those in headers and other contexts), you can set the option -v=Words; using -v=Words+math, the equation contents will be included (but not the enclosing $...$).

2.6 Summary information

By default, TEXcount outputs counts of text words, header words, caption words, number of headers, number of floats/figures, number of inlined formulae, and number of displayed formulae, and lists each of these counts. To shorten this to a one-line format per file, specify -brief.

To get TEXcount to produce a total count, specify -sum: this will compute the sum of all words plus the number of formulae. A customized sum may be computed by speficying -sum=n,n,... with up to seven numbers separated by commas giving the weight (0=don't count, 1=count once) of each of the seven counts: e.g. the default is equivalent to -sum=1,1,1,0,0,1,1. To count words only, use -sum=1,1,1. Higher weights may also be used, e.g. to count displayed formulae or floats/figures as a given number of words.

Specifying -sum has two main effects: the cumulative sum is added to the output in verbose formats, and the sum is added to the summary. If combined with -brief, the option -total is automatically set, resulting in a one line output containing only the total sum.

For adding subcounts e.g. by sections, the option -sub (or -subcount) may be used. By default, this produces subcounts by part, chapter, section and subsection which are listed in a brief format. One may, however, specify -sub= followed by part, chapter, section, or subsection (default when given without value). Break points which initiate a new subcount may also be specified within the LATEX document using %TC:break name.

If included files are included in the count (-inc), counts per file will be produced followed by a total count. Note that the counts for the included files are not included in the counts for the main document, and in particular is not included in the subcounts (e.g. per section). To suppress per file counts, the option -total may be used.

By adding the option -freq, TEXcount will output the word frequencies in order of descending frequency: this is only done for the total count, not per file. You may restrict the frequency table to words occurring at least n times by specifying -freq=n. TEXcount will count words irrespective of case, but the output will retain upper case where this is consistently used. Note that TEXcount may not recognise that words are the same if they are written differently in the code, e.g. Upper and Upper.

A frequency table for each script type (alphabetic, Han, etc. or script classes like Greek, Hebrew etc. if specified in -alphabets=) is produced by the option -stat.

2.7 Parsing options

TEXcount uses regular expressions to identify words and macro options. By default, these have been set so as to fit most common usages. However, some users may find the default to be too strict, e.g. not recognise options that are long and contain less common symbols. More permissive patterns may be selected by using the option -relaxed. This allows more general document elements to be identified as words or macro options, which may sometimes be desired, but may also have undesirable effects, so check the verbose output to verify that TEXcount has counted the appropriate elements. Conversely, if the default settings tends to combine words that should be counted as separate words, you may try the option -restricted.

Macro options, i.e. [...] after macros and macro parameters are ignored. Since TEXcount has no specific knowledge of which macros take options, this is a general rule applied to all macros that take parameters[4]. In order to avoid that uses of [...] that are not macro options are mistaken as such, TEXcount makes some restrictions on what may be contained in such an option. By default, this restriction is relatively strict under the assumption that it is better to count a few macro options as words than risk large fragments of text to be ignored. However, if your document contains macro options with more complicated values (e.g. certain special characters or macros), using -relaxed may help handle these correctly.

By default, TEXcount does not allow special characters or macros to be part of words. This may cause problems if character modifiers or some special characters are used which are entered as macros. The -relaxed option makes the word recognition regular expression somewhat more general.

2.8 File inclusion

By specifying -inc or -merge, TEXcount will automatically count documents that are included using \input or \include. The difference between the two is that -inc analyses the included files separately, while -merge merges the included documents into the parent document. Thus, -inc will result in one count per file and a total sum at the end, while -merge will treat the merged document as if it was one file.

The default option is -noinc indicating that included documents are not counted.

Paths can absolute or relative. Relative paths are by default relative to the working directory, although e.g. the import package can cause files to be included from other directories. The working directory is by default set to be the current directory: i.e. the directory from which TEXcount is executed. This default behaviour corresponds to the option -dir=.

The working directory can be specified explicitly by the -dir=*path* option. The file names on the command line should still be relative to the current directory, i.e. the one from which TEXcount is executed, while files included within these will be relative to the specified working directory.

Alternatively, if -dir is used without setting a path, the working directory is determined by the directory containing the top level LATEX documents, i.e. the document specified on the command line; if several files are provided on the command line, these may result in different working directories. Note that -dir and -dir= are fundamentally different: the first indicates that the working directory is determined by the top level LATEX documents, while the second fixes the working directory to be the current directory.

Note that when included documents are parsed as separate files, i.e. using -inc, the text of included documents is not included where the \input or \include is located. This has two consequences. First, since word counts are produced per file, subcounts, e.g. by chapter, will only include the text in the same file, not that of the included file. Secondly, if TC-instructions to TEXcount are embedded in the LATEX document, e.g. defining additional macro handling rules, these take effect in the order they are parsed by TEXcount. Since included documents are parsed after the parent document, definitions in the parent document will be in effect for the included documents; definitions made in the included documents will only be in effect for subsequently included documents, not in the parent or previously included documents.

In addition to the -dir option for setting the working directory, there is a similar option -auxdir for setting the path to the auxilary directory where e.g. the bibliography .bbl file should be read from. The default setting is -auxdir which means that working directory is used. However, -auxdir=*path* can be used to overrule this and set an alternative path. If -dir=*path* is used, the auxilary path should be relative to the current directory, not to the working directory specified with -dir=*path*; if -dir is used, as is the default, the working directory will be the directory containing the top level LATEX dodcuments (the ones specified on the command line), and the auxilary path will be relative to this, unless an absolute path is specified.

3 Macro handling rules

A few special macro handling rules are hard-coded into the TEXcount script: i.e. the handling of those can only be changed by editing the script. However, TEXcount primarily relies on a few general rules and macro and environment handling rules that follow a specific pattern.

[4] For macros that take no parameters, [...] is not interpreted as a macro option. While slightly inconsisten, this avoids e.g. {\bf[bold text]} to be gobbled up as a macro option and ignored

3.1 General macro handling rules

The general macro handling rules fall into a few general categories:

Macro In its simplest form, this type of rule just tells how many parameters to ignore following the macro. More generally, one may specify the number of parameters a macro takes and how each of these should be handled. Options enclosed in [] before, between and after parameters are also ignored; this also applies to macros not specified here, so for a macro with no rule, immediately following []-options will be ignored. (This type of rule was called an exclude rule in older versions of TEXcount, the reason being that the rule originally only gave the number of parameters to ignore following a given macro.)

Environment For environments enclosed by \begin{*name*} and \end{*name*}, there are rules specifying how the contents should be interpreted. A macro rule is added for begin*name* (without the backslash!) which is TEXcount's internal representation of \begin{*name*}. Note that special characters like * may be part of the environment name, e.g. as in equation* and rules for these need be specified[5]. *Previously, environment rules were referred to as 'group rules', but I have now renamed this both in the TEXcount script and documentation, and replaced group by envir wherever appropriate.*

Macroword This type of rule indicates that the macro itself represents one or more words. Initially, \LaTeX and \TeX are defined with values 1 indicating that each represents one word.

Preamble A few macros should be counted even if they are in the preamble. In particular, \title{title text} is counted as a header assuming it will later be used to produce a title.

Float inclusion Within floats (environments with the float parsing rule) there may be texts that should still be counted: in particular captions. These are specified with the float inclusion rule.

Previously, there was also a separate header handling rule, but this is now incorporated into the more general macro handling rules.

A macro parameter is normally on the form {*something*}; more generally it may be anything TEXcount parses as a single unit (or token), e.g. a macro, but since TEXcount parses word by word rather than character by character this may not always be correct if parameters are not {}-enclosed or macros. In addition, some macros take optional parameters which are usually on the form [*option*], and TEXcount can also (from version 4) count these.

3.2 Special macro handling rules

Some macros do not follow the pattern used by TEXcount to represent macro handling rules. For some of these, special handling rules have been hard-coded into the TEXcount script. For some, the macro syntax differs from the general rule, while in other cases the macros may trigger special processing.

file include If -inc is specified, included files will also be parsed and the total presented at the end. Initially, \input and \include trigger file inclusion, but more file inclusion macros may be added to the %TeXfileinclude hash. In addition to potentially triggering file inclusion, the syntax may differ in that input does not require the file name to be enclosed in {...}.

package include When packages are included using \usepackagename, TEXcount will check for package specific macro handling rules to include. Initially, only \usepackage triggers package inclusion, but more macros may be added to the %TeXpackageinc hash.

Complete LATEX documents should start with a \documentclass specification, then a preamble region which should not contain typeset text, before the main document starts with \begin{document}. However, LATEX files which are ment to be included into a document will not contain \documentclass and \begin{document}. A rule to recognise the preamble region is hard-coded into TEXcount.

[5] Previously, trailing * was supposed to be ignored so the same rule would apply to environment equation* as to equation. However, due to a bug in a regular expression, this did not work as intended and I have decided not to follow that strategy and instead speficy these rules explicitly.

Rules for identifying $...$, $$...$$, \(...\), and \[...\] as formulae are hard-coded and basically parse until the closing token is encountered.

The macros \def and \verb have hard-coded rules since these do not follow the pattern for macro handling rules, but may contain LaTeX code which could seriously disrupt the parsing, e.g. by containing unclosed \begin. Macros like \newcommand, however, are handled by ordinary macro rules.

The macro \biblography is handled to check if the bibliography file should be parsed. The thebibliography environment is also handled differently, one difference being that a bibliography header is added to the count.

3.3 Package specific macro handling rules

Starting with version 2.3, TeXcount can handle different sets of macro handling rules for different packages. When a package is included in the LaTeX code or through the -incpackage option, rules defined for the given package is added.

Note that TeXcount is still doing the analyses sequentially. It is therefore critical that the package inclusion takes place before any use of the package which may make a difference if you are analysing several files. E.g. if the main file contains \input setup, any packages included in setup.tex will not apply to the main file since this is parsed before TeXcount parses setup.tex.

As of now, the package support is sparse since most macro handling rules have been included in the main set of rules.

3.4 Bibliography handling

By default, the bibliography is not included in the word count. If the -incbib option is specified, however, bibliography parsing is turned on. If the bibliography is included from the bbl file using the \bibliography macro, this will be parsed as if included with the -inc option. If -merge is specified together with -incbib, the bibliography will be merged into the document.

Note that bibliography parsing may be non-trivial and depend on the bibliography style used, so the verbose output should be checked: some styles perform considerable formatting which may confuse TeXcount. In addition, initials, page numbers, etc. will all be counted as words, which may result in a word count which is higher than intendet.

3.5 Adding or modifying macro handling rules

There are basically two different ways in which you can add additional macro handling rules, e.g. for your own macros, or modify existing rules: by modifying the TeXcount script, or by adding the rules through TeXcount instructions embedded in the LaTeX code.

The simplest method is to use TeXcount instructions which are embedded in your LaTeX document as LaTeX comments on the format %TC:*instruction*. This approach is described in some detail in section 5.1.

It is also possible to modify the TeXcount code. The macro handling rules are mostly defined in the hash tables named TeXmacro, TeXenvir, etc., and editing these definitions is simple and does not required in-depth knowledge of Perl. A brief overview of the TeXcount code is provided in section 8.

3.6 Cautions!

Since the rules are of a relatively general nature, macros that have a great deal of flexibility are hard to deal with. In particular this applies to macros with a variable number of parameters or where the handling of the parameters are not constant.

By default, TeXcount assumes that macro options, i.e. parameters on the form [...], should not be counted. From version 4.0, TeXcount allows rules for optional parameters, but in most cases where optional parameters have not been specified in the macro handling rules, they will simply be ignored. There is some risk of misinterpreting text as an option: e.g. \bf[text]. This is not likely to be a frequent problem. However, if something like \bf[a lot of text] gets ignored because it is considered an option, it can influence the word count substantially. I have therefore been somewhat restrictive with what (and how much) may go into

an option. The default restriction on what may be allowed as an option may sometimes be too restrictive, causing TEXcount to interpret options as text or macro parameters; you may use the command line option -relaxed to relax this restriction and allow more general options.

More advanced macros are not supported and can potentially confuse TEXcount. In partcular, if you define macros that contain unbalanced \begin–\end, this will cause problems as TEXcount needs to keep track of these to know where environments start and end.

4 Output from TEXcount

TEXcount will by default provide a summary of the word and element counts. This may, however, be modified either by specifying -brief which reduces it to a one line summaryper file, -total to suppress per file summaries, or by providing an alternative template.

If there are parsing errors, TEXcount will print warnings about these. You may turn off this by specifying -quiet (-q for short), but there will still be an added comment about the number of errors in the final statistics to warn you of any errors.

4.1 Count statistics

The summary output will by default provide a summary of all counts: i.e. word counts for text, header and captions, and the number of floats/tables, headers, inlined and displayed formulae. You may combine these into a summary count by using the -sum option which by default gives the total number of words and formulae. You may choose briefer output formats by using the -brief option which produces a one-line summary of the counts. The option -1 is the same as specifying -brief -total and will give only one line of output for the total only. Combining -brief with -sum will cause only the sum to be printed rather than the full set of counts.

If multiple files are processed in one run, TEXcount will by default provide summary statistics per file. If files are included (using the -inc option), summaries of all files are provided as well as the total. If there is more than one file, i.e. main LATEX documents provided in the command line, it will also write a total summary.

In order to only write the total summary, use the option -total. If there is only one file processed, the result will be similar except that subcounts (counts per section etc.) are not provided with the total count.

4.2 Customising the summary output

You may specify an output template to use instead of the default output formats. This will replace the output per file or for the total with output produced using this template.

The template is a string with codes for inserting the count values and titles. To specify it, use the option -template="*template*". The encapsulating "..." are required if the template contains spaces. You may insert line shifts by using \n.

The counts may be included by using the counter keywords: word, headerword, etc. Other codes that may be inserted are: {SUM} to insert the count as specified by the -sum option, {TITLE} for the title (e.g. section name) and a header (same as title unless TEXcount has replaced it), {ERROR} for the number of parsing errors, {WARNINGS} for the number of distinct warnings, or {NWARNINGS} for the total number of warnings. Some of these also have shortened forms[6] like {ERR} and {WARN}.

Conditional inclusion may be performed using the format {*label*?...?*label*} where *label* is one of the counter keywords, SUM, ERROR or TITLE (or their alternative forms). The enclosed text will then be included only if the corresponding value exists and is non-zero. If you wish to include an alternative text when the value is non-existant or zero, use the format {*label*?*if non-zero*|*if zero*?*label*}.

Subcounts, e.g. per section, may be included by using {SUB|*template*|SUB} with a separate template text specified for the subcounts. This will only be included if there is more than one subcount, and in order to conditionally include prefix and suffix you may use {SUB?*prefix*|*template*|*suffix*?SUB}.

[6] Previously, one-letter versions of some of these codes were permitted, but that is no longer the case.

Note that you have to insert line shifts yourself. TEXcount will only insert one line shift after each file count, and not after the total count: if you process only one file and want only to output the total sum without a line shift at the end, use -sum -total -template="{SUM}", which should give the same output as -1 -sum when there are no parsing errors.

5 TEXcount instructions in the LATEX document

It is possible to give some instructions to TEXcount from within the LATEX document. These can be used to control the parsing of the document and add custom made macro and environment handling rules directly from the LATEX document. The general format of these instructions is

%TC:*instruction [parameters]*

which LATEX will interpret as a comment but TEXcount will detect.

Adding your own macro handling rules is relatively simple. While it is fairly easy to edit the script to add more rules, this has the disadvantage that the modifications will be lost if updating to a new version of TEXcount. A better and more flexible solution is to include instructions to TEXcount in the LATEX documents, alternatively to make a definition file in which new macro handling rules are defined. The TEXcount instructions for doing this take the form

%TC:*instruction name parameters [option]*

where in some rules name is the macro name (including backslash), and some rules use the or environment name. Note that rules that take the macro name as an argument can (usually) by applied to \begin{*name*} by specifying it as begin*name* (no backslash) which is how TEXcount represents \begin{*name*} internally.

macro ***macroname parameter-rules*** Defines macro handling rule for the specified macro. The parameter is on the form [*rule*,...] where each rule is either a keyword indicating the parsing rule for a macro parameter or option:*rule* for an optional []-enclosed parameter. Alternatively, an integer value n indicates that the n first parameters to the macro should be ignored, equivalent to giving a list of n ignore rules.

envir ***envirname parameter-rules content-rule*** (The previously used command, group, remains an alias for envir, but the name envir is more appropriate and therefore recommended.) This specifies the handling of environments with the given name. The parameter handling rule, applied to parameters following \begin{*name*}, are specified as in the macro instruction. The second parameter specifies the rule, i.e. parser state, with which the contents should be parsed.

macrocount ***macroname [count-spec.]*** (An alias for macrocount is macroword; the preferred name was changed to reflect that this can count any element, not just words.) If a number is provided as the count parameter, this defines the given macro to be counted as the specified number of words; if no count is specified, it is assumed to be 1. Alternatively, a []-enclosed list of counters can be specified (using the counter keywords), causing each of them to be incremented: counter are word/text, headerword, otherword, header, float, inlinemath, displaymath plus a number of aliases.

breakmacro ***macroname*** Specify that the given macro should cause a break point.

floatinclude ***macroname parameter-rules*** Specify macro handling rules used within float groups. The handling rules are specified as for macro. Most commonly, the parameter rule will be the otherword/oword to specify that words should be counted as *other words*.

preambleinclude ***macroname parameter-rules*** Specifies macro handling rules to be used in the preamble: the text between \documentclass and \begin{document}. The rule is specified like the macro rules.

fileinclude ***macroname file-path-spec.*** Specifies macros that cause files to be included when TEXcount is run with the -inc option. The parameters specify the format on which the file path is specified, and can also be used to modify the search path used within the included document.

Note that macro handling rules are added successively throughout the session: i.e. if more files are parsed, handling rules from previously parsed files still apply. This has advantages as well as disadvantages. If you give a list of files with the rules specified in the first file, these rules will be applied to all the documents. However, if you use the -inc option, included files will be parsed only after TEXcount has finished parsing the file in which they are included, so any rules specified in these will not apply to the initial document.

A few additional TEXcount instructions exist to control the overall parsing and counting:

break ***title*** Break point which initiates a new subcount. The title is used to identify the following region in the summary output.

incbib or **includebibliography** Sets bibliography inclusion, same as running TEXcount with the option -incbib.

subst ***macro text*** This substitutes a macro with any text. The verbose output will show the substituted text: e.g. %TC:subst \test TEST will cause a following \newcommand\test{TEST} to be changed into \newcommand TEST{TEST}, which TEXcount will interpret differently. Use with care!

ignore Indicates start of a region to be ignored. End region with %TC:endignore.

insert ***TEX-code*** Insert TEX code for TEXcount to process.

newcounter ***name [description]*** Define a new counter with the given name and description (optional). A corresponding parsing rule will also be added with the same name.

newtemplate and **template** ***[template-line]*** Specify a template for the summary output. The first line should just declare a new template using %TC:newtemplate, while the subsequent lines use %TC:template followed by text specifying the template. The line breaks in the template specification are not of importance: to specify a line break, use \n.

In addition, in part with debugging in mind, the following TEXcount instructions exist:

log ***[text-line]*** Writes a line of text to the verbose output. Counters may be included in the text using the {name} format.

assert ***[counts] [text-line]*** This takes a list of counts separated by comma, and writes the text (which may contain counters on the format {name}) if the asserted counts do not match the actual counts.

These may be subject to change at some later point, although the functionality should remain.

5.1 Parameter and content handling rules

There are a set of alternative rules that may be used for parsing macro parameters and environment contents. These rules, or *parser states*, are identified by keywords:

Text: (key: text, word, wd, w formerly 1) Count as text (i.e. count words).

Header text: (key: headertext, headerword, hword, hwd, hw formerly 2) Count as header text.

Other text: (key: otherword, other, oword, owd, ow formerly 3) Count as float/caption text.

Displaymath: (key: displaymath, dsmath, dmath, ds formerly 7) Count as displayed math formulae.

Inline math: (key: inlinemath, inline, imath, eq formerly 6) Count as inlined math formulae.

To header: (key: header, heading, head formerly 4) Count header, then count text as headertext (transition state).

To float: (key: float, table, figure formerly 5) Count float, then parse contents as isfloat (transition state).

Preamble: (key: formerly -9) Parse as preamble, i.e. ignore text but look for preambleinclude macros.

Ignore: (key: ignore formerly 0) Ignore text, i.e. do not count, but will still parse the code.

Float: (key: isfloat formerly -1) Float contents, ignore text but look for floatinclude macros.

Strong exclude: (key: xx formerly -2) Strong ignore which ignore environments, e.g. to use in macro definitions where \begin–\end need not be balanced.

Stronger exclude: (key: xxx formerly -3) Stronger ignore, handles macros as isolated tokens without handling their parameters, to use with macro definitions like \newcommand and \def.

Exclude all: (key: xall formerly -4) Ignore all, including unbalanced braces (e.g. used by %TC:ignore and the verbatim environment). This rule may be used for environment contents, but not for macro or environment parameters or options since the exclusion causes { and [to be ignored.

The keys are used to identify the rule. Environment content rules are simply specified by giving the desired key. Parameter rules are on the form [*rule*,*rule*,...] with one rule provided per parameter, where each rule is either one of the above keywords or option:*rule* to indicate an optional parameter on the form [...]; alternatively, a single integer can be provided (not enclosed in [] to indicate that the indicated number of parameters should be ignored.

The list of parser states is in order of increasing priority: i.e. when a parser state is specified as a rule for parsing a parameter or environment content, this will only take effect if it has higher priority than the current state. Thus, text within an ignored or excluded region will not be counted.

The formerly used numberic codes are listed at the end of the keyword list for each state. Before version 4, these numeric codes were used to specify parsing rules, but although these should still work, using the named keywords is highly recommended.

The transitional states indicates an incrementation of one of the counters and then change to another state: e.g. if the header rule is specified, this will first cause the header counter to be incremented, and then a change to the headerword state in which word counts are added to the header word counter.

5.1.1 Adding a macro handling rule

The TEXcount instruction for adding (or changing) a rule for how TEXcount handles a specific macro takes the form

%TC:macro *macro-name parameter-rules*

where the macro name includes the backslash and the parameter rules can be an integer or a []-enclosed list as explained above.

If a list, [*rule*,*rule*,...], of parsing rules is provided, the macro will be assumed to take this number of parameters. Each rule is either a keyword signifying the rule, or parser state, with which the parameter will be parsed, or option:*key* for optional []-enclosed parameters. Additional []-enclosed options, between or after the macro and the parameters, will be ignored.

Macro handling rules specified for macros *name* automatically apply to *name**: i.e. a * is automatically gobbled up as a macro modifier.

Here are some examples together with corresponding macro definitions:

```
%TC:macro \refnote[text,othertext]
\newcommand\refnote[2]{\textit{#1}\footnote{#2}}

%TC:macro \newsection [header,ignore]
\newcommand\newsection[2]{\section{#1}\label{sec:#2}}

%TC:macro \NB 1
\newcommand\NB[1]{\marginpar{#1}}
```

The predefined rules can easily be read off the script file: they are hash maps defined at the beginning of the script with names TeXmacro, TeXenvir, etc.

5.1.2 Adding an environment handling rule

Rules for environments may be added on the format

%TC:envir *name parameter-rules content-rule*

for parsing \begin{name}...\end{name}. The parameter rules are specified as for the macro rule and is used to process the parameters that follow \begin{name}. The content rule is a single parsing rule to use on the environment content.

```
%TC:envir theorem [] text
\newtheorem{theorem}{Theorem}
```

5.1.3 Adding rules that apply to the preamble and float contents

Within the preamble (from \documentclass to \begin{document}) and within floating objects (tables, figures, etc. parsed using the float/isfloat states), texts and macros are generally ignored. However, it is possible to specify particular macro handling rules that apply within these regions by using the preambleinclude and floatinclude TEXcount instructions. These take the same format as the macro instruction:

%TC:preambleinclude *macro-name parameter-rules*

%TC:floatinclude *macro-name parameter-rules*

It is possible for the same macro to specify different rules for preamble, floats and general use, although for most uses these should be expected to be the same.

Preamble inclusion is typically used for macros like \title that define text that should be counted although it may be placed in the preamble. Another use is that macros that may occur in the preamble, like \newcommand and may contained unbalanced \begin–\end pairs, require a stronger exclusion than the regular ignore rule even in the preamble to ensure TEXcount is not confused by these.

Float inclusion is used e.g. for captions, and the parsing rules should normally be to count texts using the otherword parsing rule.

5.2 Count macro, either as words or in other counters

Some macros, e.g. \LaTeX, generate words and should be counted as words. Other macros can generate other elements, e.g. headers or figures. Rules for counting macros can be specified as

%TC:macroword *macro number*

where the parameter is the number of words produced by the macro, or

%TC:macroword *macro* [*countername,...*]

which causes each of the counters in the list to be incremented by one (or more if listed multiple times).

The counters for counting the number of files, text words, etc. are stored in an array. In some cases, e.g. when -sum= is specified, the order of the counters in this array is used to specify the rule. However, in most cases the counters should be specified by keywords. The counters, their index number and keywords are:

0. (keys: file) Number of files.

1. (keys: text, word, wd, w) Number of words in text.

2. (keys: headerword, hword, hwd, hw) Number of words in headers.

3. (keys: otherword, oword, owd, ow) Words outside text, e.g. in floats/tables/figures.

4. (keys: header, heading, head) Number of headers.

5. (keys: float, table, figure) Number of floating environments, e.g. tables and figures.

6. (keys: inlinemath, inline, imath, eq) Number of inlined mathematics formulae.

7. (keys: displaymath, dsmath, dmath, ds) Number of displayed equations.

Examples of uses:

```
%TC:macroword \TeXcount 1
\newcommand\TeXcount{{\TeX}count}

%TC:macroword acknowledge [header,hword]
\newcommand\acknowledge{\section*{Acknowledgements}}
```

5.3 Specifying file inclusion macros

In addition to \input and \include, which are the standard LaTeX macros for file inclusion, there are packages such as import intended to enable organising files into subfolders. TeXcount, from version 3, adds some support for macros that change the path from which files are included. If the user needs to add additional file inclusion macros, the format is

%TC:fileinclusion *macro file-parameters*

where the file parameters are a comma separated list of keywords, each corresponding to a macro parameter. Available parameters are:

input: This is a special keyword to use with \input. The handling of the parameter values is as file, but the parameter itself is not required to be enclosed in {}.

file: This parameter simply gives the name of or path to a file. If the file is not found, TeXcount will append .tex and try again.

texfile: This parameter gives the name of or path to a file, but .tex will be appended, and is the rule used by \include.

dir: This parameter provides the path of a directory relative to the $workdir, and adds this to the search path before including any files. This is used with the \import macro of the import package.

subdir: This parameter provides the path of a directory relative to the current directory, and adds this to the search path before including any files. This is used with the \subimport macro of the import package.

<bbl>: This is a special keyword to use with \bibliography to specify inclusion of the bibliography file. It is different from the other keywords in that it does not take a macro parameter.

Examples showing how some existing macros, from basic LaTeX and from the import package, are defined:

```
%TC:fileinclude \input input
\input macros.tex
%TC:fileinclude \include texfile
\include{intro}
%TC:fileinclude \import dir,file
\import{supplements/}{overview.tex}
%TC:fileinclude \subimport subdir,file
\subimport{tables/}{data.tex}
```

5.4 Adding subcount break points

By specifying -sub, TeXcount can produce subcounts, e.g. per section. Alternatively, or in addition, explicit break points can be entered in the LaTeX document using the TC-instruction break. These take the form:

%TC:break *title*

A title (or name) may be given to identify the break point.

If you define new section macros or macros you wish to cause a break point, these may be specified using the TC-instruction breakmacro:

%TC:breakmacro *macro label*

This defines the given macro to cause a break point, and uses the given label to indicate the type of break (e.g. Section, Chapter, etc.).

5.5 Ignoring segments of the file

The TC-instruction ignore, later canceled by endignore, may be used to turn of all counting in a segment of the LaTeX file. The ignored segment should thus be started by

%TC:ignore

and ended by

%TC:endignore

causing all text inbetween to be ignored.[7]

5.6 Bibliography inclusion

In order to include the bibliography in the word counts, you can either specify -incbib on the command line, or use TeXcount instruction

%TC:incbib

which has the same effect: it specifies handling rules for the \bibliography macro and thebibliography environment that causes the bibliography to be included in the count, and if necessary the .bbl file to be included (without requiring -inc or -merge).

5.7 Text substitution prior to parsing

There are cases where a macro needs to be substituted with a text prior to parsing. One such case is when a macro contains a file path which is later used by a file inclusion macro. Since TeXcount does not actually expand the macros, it will not be able to generate the file path from the macro. Instead, one may perform an explicit substitution

%TC:subst *macro text*

which will then cause all occurrences of the macro to be substituted by the provided text prior to parsing. Note that this substitution will therefore also be found in the verbose output.

```
\newcommand\chappath{chapters}
%TC:subst \chappath chapters
\input \chappath/chapter1
```

Note that the substitution is placed *after* the \newcommand definition. Otherwise, the substitution would have taken effect, changing that line to \newcommand chapters/chapters.

[7] In older versions, TeXcount would still parse this text and might thus be affected by unbalanced braces. As of version 2.3, however, this should be fixed to make the ignore instruction more robust.

5.8 Adding a new counter

Initially, TeXcount has eight different counters: file, text words, header words, other words, number of headers, number of floating objects, number of inlined formulae, and number of displayed formulae. However, it is possible to add more counters, e.g. to count footnotes separately. The syntax is

%TC:newcounter *name [description]*

where the given name is used as keyword to refer to this counter. If no description is provided, the name will be used as description. A new counter is then added, and a parsing rule (parser state) with the same name is added which may be used in specifying macro and environment handling rules.

The following example shows how two different counters are added: one to count the number of footnotes, and another to count the words in footnote.

```
%TC:newcounter fwords Words in footnotes
%TC:newcounter footnote Number of footnotes
%TC:macro \footnote [fwords]
%TC:macroword \footnote [footnote]
Each footnote\footnote{Words in footnotes will be counted separately.} will be counted.
```

Note that we have to specify one rule for counting the words in footnotes, and another rule for counting the footnotes. Unlike headers and floating bodies, there are no transition states available that can do both.

6 Using an option file

If you have a lot of settings, e.g. output template and TC commands for specifying parsing rules, you may place these into a file and include this using -opt=*file*.

The format of this file is quite simple: each line is read as one option, so different options should not be placed on the same line. If some options are so long you need to break the line, e.g. for specifying an output template, you can do so by placing \ at the start of lines that continue the previous line.

You may enter TC commands just as in the LaTeX code by starting the line with % instead of TC:. Using these, you may include specifications of parsing rules.

Blank lines and lines starting with # are ignored and may thus be used to add comments to the option file. So are leading spaces, which allows lines to be indented. Line breaks may be inserted by \n.

Here is an example which sets the total sum to be the number of words (not including formulae), subcounts by section, parses included files, and adds an output template.

```
### Options to use with TeXcount

# Counting options
-sum=1,1,1
-sub=section
-inc

# Macro rules
%macro \url 1
%envir sourcecode 0 0
%macroword \TeXcount 1

# Path used in file inclusion (\chapterpath filename)
%subst \chapterpath chap/

# Output template
-template=
   \::: {title} :::\n
   \Words: {sum}\n
   \Formulae: {6} + {7}\n
   \{5?Number of floats: {5}\n?5}
   \{SUB? - {sum} words in {title}\n?SUB}
```

7 Customising TeXcount

TeXcount is a self-contained Perl script: no external packages or resources required except that you need to have Perl installed to run it. Unfortunately, as with much of Perl code since Perl does not itself encourage

structured programing, after expanding somewhat in size, it is not the most readable of codes. However, there may still be cases where you might yourself want to modify the code.

There are some things that may be modified quite easily even without knowing Perl.

Preset startup options On one of the first lines of the code, the list @StartupOptions is defined. A list is simply a sequence of values (an array) on the form (*value*,*value*,...). As it stands, this list is empty, but you may add startup options to be included prior to command line options when you run TEXcount. E.g. if you change this to ("-inc") it will automatically add the -inc option so you don't have to do that yourself every time you run TEXcount.

Adding macro handling rules While you may add macro handling rules using %TC: commands either in the document or in a separate option file, this is inconvenient for large numbers of macros or if you want these rules always to be included. Also, you might want to add such rules for specific packages. In either case, it might be practical to add these directly to the TEXcount code. TEXcount stores the rules in hashes (maps from a key to a value) named %TeXmacro, %TeXenvir, etc. There is more documentation on each of these in the code itself, and you may also inspect how rules have been defined for other macros and environments.

Output style The ANSI colour codes for different levels of verbosity are encoded in the %STYLES hashes and may be changed. The HTML style is encoded in the method html_head() and is easily modified.

Character and word definitions TEXcount identifies words as those that match one of a given set of regular expressions (defined in @WordPatterns). Note that @WordPatterns is changed by options -chinese, -japanese and -letters. The pattern that is used within the word patterns to recognise letters is stored in $LetterPattern. This is replaced if the -relaxed or -restricted option is set. Changing these definitions may be useful if you have special characters or wish to define words differently.

8 Modifying the TEXcount script

TEXcount is written in Perl, and although hardly the best structured and documented code ever seen, I have tried to structure and document it somewhat. In particular, some parts of the code should be easily modifiable even without in-depth knowledge of Perl or the TEXcount script: e.g. the macro handling rules.

For more aid on how the TEXcount script is coded and organised, please consult the Technical Documentation. However, here is a very brief overview:

Header and imports: The shebang (#!) and package imports (use *package*).

Global variables (and some methods related to these): This defines and initialised global variables related to option settings, state variables used in parsing and counting (including functions for unterpreting these), variables and hashes for storing macro handling rules, and character class definitions (must be defined before use).

Main program: This simply contains a call to the MAIN routine with the command line arguments.

Routines/functions/procedures: The first procedure defined is MAIN which contains the program flow, then follows other subroutines. Routines with capitalised initial letters indicate high-level routines, while routines starting with underscores (_) are low-level routines.

Text data: At the end of the file is a __DATA__ region containing text data used by the help routines.

Perl will first process the setup section which defines global variables, arrays and hashes. It then executes the main section (consisting of the call to MAIN), whereafter it exits. The subroutines and text data follow after the exit.

9 License

The TEXcount package—script and accompanying documents—is distributed under the LATEX Project Public License (LPPL)

http://www.latex-project.org/lppl.txt

which grants you, the user, the right to use, modify and distribute the script. However, if the script is modified, you must change its name or use other technical means to avoid confusion with the original script.