

Jan 16, 04 10:20

stdin

Page 1/12

```

Lopar                               lopar(1)
NAME
  lopar - left-corner parser for probabilistic context-free
  grammars
SYNOPSIS
  lopar grammar-file [ input-file [ output-file ] ]
OPTIONS
  -l file
    Read the lexicon from file.
  -s file
    Read start symbol frequencies from file.
  -o file
    Read open class frequencies from file.
  -O file
    Read open class frequencies of capitalised words from
    file. If this file is empty, then capitalised words
    will not be treated differently from lower case words.
    Otherwise, the frequencies in the file argument of the
    -o option pertain to lower case words only.
  -t file
    Train the parser and store the resulting parameters in
    file.gram, file.lex and file.start.
  -viterbi
    Print Viterbi parses (one tree per line).
  -viterbil
    Print Viterbi parses (one node per line).
  -tagging
    Print best tag sequence.
  -chunking file
    Do chunking rather than parsing. The chunk categories
    are read from file. This algorithm selects and prints
    the chunk sequence where the partial tree formed by the
    dominating nodes has maximal probability.
  -chunk-trees file
    Like -chunking but prints the partial parse tree rather
    than the chunk sequence.
  -opt-chunking file
    This chunking option selects the chunk sequence for
    which the sum of probabilities of all parse trees con-
    taining exactly this chunk sequence is maximal. (It
    sums over more analyses than the -chunking option.)

```

Jan 16, 04 10:20

stdin

Page

```

SunOS 5.8                               Last change: February 1999                               1
Lopar                                     lopar(1)
-max-chunks
  Print maximal chunks rather than minimal chunks
  (requires -chunking)
-headed-chunks
  Print chunks with lexical heads (requires -opt-
  chunking).
-labelled-chunks
  Print labelled chunks (requires -opt-chunking).
-recursive-chunks
  Also print chunks containing other chunks (requires
  -opt-chunking).
-isym file
  When the parse first generated by lopar is matched with
  partially bracketed input, arbitrary additional consti-
  tuents are allowed. Therefore, the parses (VP V NP PP)
  and (VP V (NP NP PP)) are compatible. The -isym option
  restricts the set of insertable symbols to those listed
  in the argument file.
-bsmatch file
  This option provides a means to translate symbols in
  the bracketed input to grammar symbols allowing the
  symbols in the bracketed input and the grammar to be
  different. Each line of the argument file contains a
  bracket symbol followed by a sequence of grammar sym-
  bols. Here is an example:
  NP NP/nom NP/gen NP/dat NP/acc
-merge file
-m file
  Add the frequency counts from file.* to the frequencies
  of the input grammar. This option may be used more than
  once. The resulting grammar files are written to files
  whose base name is given as the input-file argument,
  (i.e. as the second argument or the first argument if
  the option -in was used.
-inside
  Print the inside probabilities of constituents and
  edges.
-outside
  Print the outside probabilities of constituents and
  edges.

```

Jan 16, 04 10:20 **stdin** Page 3/12

```

-freq
SunOS 5.8      Last change: February 1999      2
Lopar                                lopar(1)
    Print the estimated frequencies of constituents and
    edges.
-viterbi-probs
    Print the Viterbi probabilities of constituents and
    edges.
-governors
    Print governors.
-heads
    Print lexical heads.
-lexmodel
    Use a head-lexicalized probability model for training.
    This option implements the hypar functionality.
-pooling file
    Read information about parameter pooling for parent
    categories from file.
-cpooling file
    Read information about parameter pooling for child
    categories from file.
-fixed file
    Read information about fixed lexical choice parameters
    from file.
-analysis-count file
    Store the number of analyses of each sentence in file.
    Counts larger than MAX_LONG are not reported correctly.
-lsc file
    Replace lexical choice probabilities with lsc probab-
    ities from file.
-print-probs file
    Print the probability parameters used in each parse
    forest to file.
-fails file
    Print sentences which failed to be parsed to
-print-model
    Print the parameters of the probability model.

```

Jan 16, 04 10:20 **stdin** Page

```

-store-params
    Write the model parameters in binary form to a file.
-params file
    Read the model parameters in binary form from file.
SunOS 5.8      Last change: February 1999      3
Lopar                                lopar(1)
    file must have been generated using the -store-params
    option. Binary parameter files load much faster!
-vpf Print parse forests in vpf format. This option is
    needed if the parse forest is to be viewed with vpf.
-tgrep
    Substitute characters (like parentheses) which are
    problematic for tgrep.
-iter n
    Reestimate the probabilities after 1n, 3n, 7n etc. sen-
    tences (requires option -t).
-old-freq w
    Set the weight of the frequencies from the last itera-
    tion to w (requires options -t and -iter).
-rob-start file
    Read the weight for those start categories which are
    robustness categories and the robustness start
    categories themselves from file. The start frequencies
    of the robustness categories are multiplied by the
    weight before probabilities are estimated.
-in file
    Reads the grammar file from file.gram, the lexicon file
    from file.lex, the open class files from file.oc and
    file.OC and the start frequency file from file.start.
-stems
    Use stems rather than words as lexical heads. This
    option requires a different lexicon format in which
    tag/frequency pairs are replaced by tag/frequency/stem
    triples. The stems may contain blanks and must be fol-
    lowed by a tab or newline character.
-symbolic
    expect purely symbolic grammar files without frequency
    information.
-lattice-input
    The parser reads word lattices rather than word

```

Jan 16, 04 10:20

stdin

Page 5/12

```

sequences. Each input token is preceded by its start
and end position.
-high-precision
  Use increased precision to print numeric values to text
  files.
-quote
  Quote strings in the parser's output.
SunOS 5.8          Last change: February 1999          4
Lopar              lopar(1)
-sent-probs
  Print the inside probability (Viterbi probability in
  case of Viterbi parsing) of the whole sentence to
  /dev/stderr.
-shave t
  This option removes grammar rules whose frequency is
  lower than t.
-preorder
  Print parse forests in prerequisite order.
-textpar
  Print the lexicalized parameters in text form to addi-
  tional files with extension txt. These files allow
  manual inspection of the frequencies.
-trees
  Print sequences of parse trees rather than parse
  forests.
-no-smoothing
  Don't do any parameter smoothing.
-sent-weights
  Each sentence is preceded by a sentence weight. This
  might be useful if sentences are often repeated.
-interactive
  Interactive mode for examination of parameters.
-verbose
  Turn on verbose mode.
-version
  Print the programs version number.
-quiet
  Suppress most parser messages.

```

Jan 16, 04 10:20

stdin

Page

```

-timing
  Print the processing time for each sentence.
-rob-rules file
  Read the weight for the robustness rules and the
  robustness rules themselves from file. The rule fre-
  quencies of the robustness rules are multiplied by the
  weight before probabilities are estimated. This option
  was intended to result in a preference for regular
  rules, but it didn't work as well as expected.
-min-start-freq f
SunOS 5.8          Last change: February 1999          5
Lopar              lopar(1)
  Only store lexicalised start frequencies larger than f.
  The default is 0.1.
-min-rule-freq f
  Only store lexicalised rule frequencies larger than f.
  The default is 0.5.
-min-choice-freq f
  Only store lexical choice frequencies larger than f.
  The default is 0.5.
-wordwise-smoothing
  Do word-wise smoothing. Normally lexical rule probabil-
  ities for unlexicalised grammars are smoothed by
  discounting the tag/word frequencies for each tag and
  then distributing the discounts uniformly over unseen
  words. When the option -wordwise-smoothing is speci-
  fied, the discounting is done word-wise with a non-
  uniform backoff distribution. The backoff distribution
  is derived from the tag frequencies of all words with
  the same set of possible tags.
DESCRIPTION
  lopar is a parser for probabilistic context-free grammars
  (see FILE FORMATS). It is able to do purely symbolic pars-
  ing, probabilistic parsing, lexicalised probabilistic pars-
  ing, tagging, chunking and parameter estimation with and
  without parameter pooling.
  lopar expects three arguments. If only two arguments are
  specified, the output is directed to standard output. If
  only one argument is specified, input is read from standard
  input.
  By default, the resulting syntactic analyses are printed in

```

Jan 16, 04 10:20

stdin

Page 7/12

parse forest representation (see FILE FORMATS). The options *-inside*, *-outside* and *-freq* turn on the output of inside or outside probabilities or estimated frequencies, respectively. These options require that a lexicon is specified with the *-l* option. The parser will abort if the grammar is not cycle-free.

If one of the options *-viterbi* and *-viterbi1* is specified, Viterbi parses will be printed rather than parse forests. (The options *-inside*, *-outside* and *-freq* have no effect here.)

If the option *-tagging* is specified, the best sequence of tags will be printed. If *-viterbi* is specified as well, then the tags of the best parse are printed. Otherwise the tags where inside probability times outside probability is

SunOS 5.8 Last change: February 1999 6
Lopar lopar(1)

highest are printed. (The options *-inside*, *-outside* and *-freq* have no effect here.)

If the option *-chunking file* is specified, the best chunk sequence will be printed. The definition of the set of chunks is read from *file*. Each line of the parser one sentence. Each chunk is represented by its category, start and end position (starting with 0) enclosed in parentheses, followed by that part of the input sentence which is covered by the chunk (enclosed in double quotes).

The option *-heads* turns on head-lexicalized parsing.

The options *-iter* and *-old-freq* modify the training scheme. If *-iter* is specified, the parser will reestimate the probability model in intervals of exponentially growing length. If *-old-freq* is specified, the old frequencies are multiplied by the weight factor after reestimation. Otherwise they are set to zero.

FILE FORMATS

The *grammar* file contains one grammar rule per line. Each grammar rule starts with its frequency followed by the mother category (symbol on the left-hand side) and the daughter categories (symbols on the right-hand side). There is a Perl script called *galacsy2lopar* which converts Galacsy style grammars to lopar format.

The *lexicon* file contains one lexicon entry per line. Each

Jan 16, 04 10:20

stdin

Page

lexicon entry starts with the word (which may contain blanks) followed by a tab and a sequence of part-of-speech/frequency pairs.

Words containing blanks are interpreted as multi-word lexemes and they are automatically recognised in the parser input. Given the lexicon entry

```
New York City NP
```

the parser will correctly recognise this multi word expression in the input

```
He
moved
to
New
York
City
```

SunOS 5.8 Last change: February 1999 7
Lopar lopar(1)

Each line of the *start symbol* file contains a grammar symbol and its frequency as root of parse trees.

Each line of the *open class* file contains a grammar symbol and its frequency as category of unknown words. This information is needed for parsing unknown words. The *parameter pooling* file specifies which mother categories are to be pooled in the lexical choice distribution of the lexicalised PCFG. Each line corresponds to one pooling class. It starts with the name of the pooling class which is followed by the names of the categories whose parameters are to be pooled.

Each line of the *fixed choice parameter* file contains a lexical head, a (pooling) category, a mother (pooling) category and a mother node head. The parser will set the corresponding lexical choice probability to 1. This is useful to ensure that German verbs with separable particles will choose the correct particle. In order to ensure that the verb *wartet* with lexical head *abwarten* chooses *ab* as particle, the following line should be added to the fixed parameter file:

```
ab PART VP abwarten
```

The *input* text which is to be parsed has to be in one-word-per-line format. Each sentence must be followed by an empty line. If no lexicon is given, the tags have to be provided

in the input after the word. Only symbolic parsing is possible without a lexicon, however. Each tag has to be preceded by a tab character. Words may contain blanks. If a lexicon was specified, then the tags are optional. The input may contain (partial) bracketing. Each opening (or closing bracket) is on a separate line which starts with a tab character. Opening brackets are optionally followed by a list of category symbols. The parser eliminates all analyses which are incompatible with the input bracketing either because of crossing brackets or because of category mismatches.

Example:

```
I
  ( VP
    saw
    the
    man
  )
  on
  (
    the
    hill
    with
    the
```

SunOS 5.8 Last change: February 1999 8
Lopar telescope lopar(1)

Parsing of word lattices (as produced e.g. by a speech recogniser) is possible, as well, by using the parser option `-lattice-input`. Lattice input requires that each input token is preceded by its start and end position. Here is an example.

```
0 4 recognise
0 1 wreck
1 2 a
3 4 nice
4 5 speech
4 5 beach
```

The output representation of a *parse forest* contains one constituent per line. The line number (starting with 0) is also the index of the constituent. A constituent starts with the category name followed by its start and end position. If one of the options `-heads`, `-inside`, `-outside` or `-freq` is specified, then the position of the lexical head or the inside or outside probability or the estimated frequency

follows, respectively. Then follows information about the analyses of the constituent. In case of terminal nodes, an analysis consists of the word. In case of non-terminal nodes, it consists of a sequence of analyses which are separated by percent signs. Each analysis consists of the rule number (corresponding to the position of the rule in the grammar file), followed by the inside or outside probability or estimated frequency of the edge if one of the respective options was specified, and the indices of the daughter constituents. The line ends with two percent signs.

EXAMPLES

Many combinations of parameters make sense for `lopar`. Here are a few examples.
The following command will parse *file* with *grammar* and print the chart contents. The input of the parser must be tagged because no lexicon is specified.
`lopar grammar file`
This command will train the tagger and write the parameters to the files `out.gram`, `out.lex`, `out.oc`, `out.OC` and `out.start`.

`lopar grammar -l lexicon file -t out`
The next command uses a different training scheme. The parameters are reestimated after 100, 300, 700 sentences and

SunOS 5.8 Last change: February 1999 9
Lopar lopar(1)

so on. Unparsed sentences are written to the file *unparsed*.

`lopar grammar -l lexicon file -t par -iter 100 -old-freq 0.9 -fails unparsed`
The following command will parse *file* and print the chart contents with inside probabilities. The input may be untagged here. Output is written to the file *result*.

`lopar par.gram -l par.lex -s par.start -inside file result`
This command will print viterbi parses in one-word-per-line format. The option `-in par` is used here instead of `par.gram`
`-l par.lex -o par.oc -O par.OC -s par.start`.

`lopar -in par -viterbil file`
The next command tags its input.

Jan 16, 04 10:20

stdin

Page 11/12

Jan 16, 04 10:20

stdin

Page 1

University of Stuttgart, Email: schmid@ims.uni-stuttgart.de,
 All Rights Reserved
 SunOS 5.8 Last change: February 1999 11

cat file | lopar -in par -tagging
 This command also tags but uses the Viterbi rather than the Inside-Outside algorithm.

lopap -in par -tagging -viterbi file
 The next command does unlexicalized training, but writes lexicalized parameters to the files *par.lgram*, *par.lchoice* and *par.lstart*. This command is comparable to *ultra*. It generates lexicalised frequencies from unlexicalised parameter files.

lopap -in par -t out-par -heads file
 This command does lexicalisation with parameter pooling.

lopap -in par -t out-par -pooling par.pool -heads file
 This command does lexicalized training and expects lexicalized parameter files as input (comparable to the *hypar* program).

lopap -in par -t out-par -lexmodel file
 The next command makes use of fixed lexical choice parameters.

lopap -in par -t out-par -lexmodel -fixed par.fix file
 Lexicalized parsing with lemmas as lexical heads is done with this command. (The output is written to *file2*.)

SunOS 5.8 Last change: February 1999 10

Lopap lopap(1)

lopap -in par -viterbil -stems -lexmodel file file2

EXIT STATUS

lopap returns 0 unless some error occurs.

BUGS

Some of the less frequently used options have not been tested systematically and might be buggy. The same holds for unusual combinations of options.

SEE ALSO

lopap-charge, vpf

AUTHOR

Helmut Schmid, Institute for Computational Linguistics,