NAME

bitpar − A parser for treebank grammars with traces

SYNOPSIS

bitpar *grammar lexicon* [*infile* [*outfile*]]

OPTIONS

−o	Print parse forests. (By default, BitPar just parses the input without generating output.)
−s sym	Use the symbol *sym* as start symbol rather than the first symbol of the grammar file.
−p	This option tells the parser that the grammar and the lexicon contain frequencies. Without this option, the parser expects a symbolic grammar rather than a PCFG. Many other options implicitly set the -p option: -v -ip -vp -f -b -em
−v	Print the Viterbi parse (implies option -p).
−u file	Read possible part-of-speech tags of unknown words from *file.* Each line of the file contains a tag and a frequency.
−w file	The deterministic finite state automaton stored in *file* is to be used for the classification of unknown words. The parser estimates POS tag probabilities for each class of unknown words from the POS tag probabilities of the words in the lexicon belonging to the same word class. Only POS tags listed in the argument file of option -u are possible tags of unknown words.
−S w	Sets the wordclass smoothing weight to w. The default is 1. The parser smooths the POS tag probabilities of a word by adding the tag probabilities of the corresponding word class weighted with the factor w to the frequencies of the word in the lexicon file.
−tg	This option has to be used if the grammar contains rules of the form VP V *NP* where *NP* is a trace symbol which (implicitly) expands to the empty string. The parser ignores the trace symbols during parsing and inserts them when the parse is printed.
−ts xy	This option specifies that trace symbols are enclosed by the characters x and y rather than *. The option "-ts ’()’" specifies e.g. that trace symbols are of the form (NP) or (NP(-NONE-(*))).
−H	One node of each grammar rule is marked as the head by a preceding ˆ symbol.
−l	The lexicon contains lemma information. The lemma follows the POS tag (and the frequency if present).
−a f	read a list of association scores from the file f. This option implies lexicalized parsing, i.e. computation of a parse forest, pruning with a low threshold, and assignment of a head to each constituent. Many nodes have to split in order to make the lexical head unambiguous. The lexicalized parse probability is defined as the unlexicalized probability multiplied by the association scores for each word and its governor. The governor of a word is the lexical head of the nearest ancestor node which is not headed by the word. (See below for the file format.)
−b n	Print the n most probable parse trees rather than a parse forest (implies option -p).
−vp	Print parse forests or n-best parses with Viterbi probabilities (implies option -p).
−ip	Print parse forests with inside probabilities (implies option -p)
−f	Print parse forests with estimated frequencies (implies option -p).
−em f	Do EM training using the inside-outside algorithm. f ist the prefix of the files, where the output is stored.
−t	Print trace probabilities (implies options -p and -tg)
−prune t	Eliminate edges with an estimated frequency below the threshold t. (The estimated frequency of an edge is the product of the outside probability of the parent multiplied by the inside probability of the edge and divided by the inside probability of the root.) This option implies option -p.

−t Print trace probabilities.

−rn Print parse forests with rule numbers.

−mf Print parse trees with the highest estimated f-score.

−y Print parse forests in YAP format. This option is required if bitpar is to be used as context-free parser in the YAP system.

−q Suppress status messages

−i verbose mode

−h Print information about program usage.

FILE FORMATS

The *grammar* file contains one grammar rule per line. Each grammar rule starts with its frequency (unless the grammar is purely symbolic) followed by the parent category (symbol on the left-hand side) and the child categories (symbols on the right-hand side). The symbols are separated by whitespace. The first symbol in the grammar file is the start symbol (unless option -s is used). If the option -H is used, one of the child nodes has to be head-marked with a preceding ˆ symbol.

The *lexicon* file contains one lexicon entry per line. Each lexicon entry starts with the word (which may contain blanks) followed by a tab and a sequence of part-of-speech tags. If the option -p is used, the POS tag must be followed by whitespace and a frequency value. If the option -l is used, it has to be followed by whitespace and a lemma.

Example:

```
saw     NN 3 saw        VB 1 saw        VBD 9 see       VBP 1 saw
```

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. Words may contain blanks.

The *association score* file contains one entry per line. Each entry consists of a word w, a governor g, the word’s part-of-speech tag C, and the score. The four fields are separated by whitespace. The association score is closely related to the pointwise mutual information. It is defined as the joint probability p(w,g|C) of the word and its governor (given C) divided by the product p(w|C)p(g|C) of the marginal probabilities of the word and its governor (given C). There are also *default score* entries (where the special string <DEFAULT> replaces the word). The default scores are assigned to all word-governor-POS tag triples which are not explicitly listed in the table. Triples to which neither an explicit entry nor a default entry is applicable get an association score of 1.

Example:

```
vice        president       NN      170.555216771753
executive       president       NN      29.2011388067248
<DEFAULT>       president       NN      0.10849721998647
```

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. Words may contain blanks.

The *word class automaton* file has the following format: Lines corresponding to state transitions have the form *<state1><tab><char><tab><state2>* where <state1> and <state2> are numbers indicating the start and the target state of a transition, <tab> is a tabulator character and <char> is the character which is consumed by the respective transition. Lines indicating final states have the form *<state><tab>wordclass<tab><class>* where <state> is a state number, and <class> is the number of the respective word class. State numbers start at 0 and word class numbers start at 1. The transitions are sorted by (i) increasing start state number and by (ii) the transition symbol (encoded as unsigned char). The lines corresponding to the final states are at the end of the file and they are sorted by increasing state number.
Here is an example automaton which assigns word class 1 to sequences of a’s and word class 2 to sequences of b’s and wordclass 3 to mixed sequences of a’s and b’s.

```
0       a       1
0       b       2
```

```
1    a          1
1    b          3
2    a          3
2    b          2
3    a          3
3    b          3
1    wordclass  1
2    wordclass  2
3    wordclass  3
```

EXIT STATUS

bitpar returns 0 unless some error occurs.

SEE ALSO

vpf

AUTHOR

Helmut Schmid, Institute for Computational Linguistics, University of Stuttgart, Email: schmid@ims.uni-stuttgart.de, All Rights Reserved