# MICA Parser
# Documentation of The Output DEPS Format

Owen Rambow
rambow@cs.columbia.edu

## 1 Introduction

This is the documentation of the "DEPS format", which is the output of the MICA parser. The MICA parser is a fast, freely available parser based on Tree Adjoining grammar [Bangalore et al., 2009]. The MICA parser is available at `http://mica.lif.univ-mrs.fr/`.

The DEPS format is used to represent dependency trees: each line represents one dependency link, and information associated with the node at the lower end of that link. This format has evolved over many years. Many people have contributed to it, principally Srinivas Bangalore (who originated it) and John Chen. Owen Rambow added additional features. Owen Rambow is the author of this documentation, not the sole creator of the format.

## 2 Basic Format

Each line is one dependency link. The nodes of one tree follow one another. The order does not matter. Nodes from different trees are marked at the boundary by two lines of `...EOS...//...EOS...//...EOS...`. Right before the first line of `EOS`, the log of the probability of the derivation if printed as follows:

```
# lp=-43.412087\verb
```

Note that if anything follows the number, you should ignore it. Different input sentences are marked at the boundary by two lines of `...NEW...//...NEW...//...NEW...`. This means that if you have chosen only 1-best output, the tree for each sentence will be delimited by first two lines of `EOS`, then two lines of `NEW`. If you have chosen $n$-best output, then the different analyses will be given in order best to worst (delimited by `EOS`).

Each line comprises the following fields, separated by a single space:

ID word POS parent-ID parent-word parent-POS supertag parent-supertag difference `||` attributes

These fields have the following meaning:

- ID: a number unique to this sentence representing this node. The nodes are shown in order of the ID. The IDs represent surface order.
- word: this is the word form output by the tokenizer, if the tokenizer was used, or the input word form, if no tokenizer was used.
- POS: this is the Penn Treebank part-of-speech tag (see `http://www.comp.leeds.ac.uk/amalgam/tagsets/upenn.html`.
- parent-ID: ID number of parent node (necessarily in same sentence!).
- parent-word: the word of the parent. This is redundant in the sense that it can be retrieved from the parent ID; however, it is repeated here so that grep can be easily used on these files.
- parent-POS: the Penn Treebank part-of-speech tag of the parent. This is redundant in the sense that it can be retrieved from the parent ID; however, it is repeated here so that grep can be easily used on these files.
- supertag: supertag of this node.
- parent-supertag: supertag of parent.
- difference: parent-ID minus ID.
- attributes: a list of the form `attribute>:`value¿*. Note that there is always a value. The list of possible attributes can vary depending on the properties of the word. The attributes are described in Section 4. Note that these are only output if the `-R` option is used when invoking MICA.

Example:

1 Time NNP 2 flies VBZ t3 t81 2——cat:N dsubcat:nil ssubcat:nil pred:- comp:n root:NP lfront:nil rfront:nil intern:nil adjnodes:NP substnodes:nil DSub:nil SSub:nil DRIO: SRIO: DRole:0 SRole:0
2 flies VBZ 2 flies VBZ t81 t81 0——cat:V dsubcat:NP0 ssubcat:NP0 voice:act comp:n root:S lfront:NP#s#0 rfront:nil intern:VP adjnodes:S_VP substnodes:nil DSub:0 SSub:0 DRIO:0 SRIO:0 DRole:Root SRole:Root
3 like IN 2 flies VBZ t13 t81 7——cat:IN dsubcat:NP1 ssubcat:NP1 pred:- comp:n root:PP lfront:nil rfront:NP#s#1 intern:nil modif:VP dir:RIGHT adjnodes:PP_VP substnodes:nil DSub:1 SSub:1 DRIO:1 SRIO:1 DRole:Adj SRole:Adj
4 an DT 5 arrow NN t1 t3 1——cat:D dsubcat:nil ssubcat:nil comp:n root:D lfront:nil rfront:nil intern:nil modif:NP dir:LEFT adjnodes:NP substnodes:nil DSub:nil SSub:nil DRIO: SRIO: DRole:Adj SRole:Adj
5 arrow NN 3 like IN t3 t13 5——cat:N dsubcat:nil ssubcat:nil pred:- comp:n root:NP lfront:nil rfront:nil intern:nil adjnodes:NP substnodes:nil DSub:nil SSub:nil DRIO: SRIO: DRole:1 SRole:1
6 . _PERIOD_ 2 flies VBZ t26 t81 8——cat:. dsubcat:nil ssubcat:nil

comp:n root:. lfront:nil rfront:nil intern:nil modif:S dir:RIGHT adjnodes:S substnodes:nil DSub:nil SSub:nil DRIO: SRIO: DRole:Adj SRole:Adj
# lp=-51.379112

Another example:

1 I PRP 2 know VBP t29 t28 2——cat:G dsubcat:nil ssubcat:nil comp:n root:NP lfront:nil rfront:nil intern:nil adjnodes:NP substnodes:nil DSub:nil SSub:nil DRIO: SRIO: DRole:0 SRole:0
2 know VBP 2 know VBP t28 t28 0——cat:V dsubcat:NP0_S1 ssubcat:NP0_S1 voice:act comp:n root:S lfront:NP#s#0 rfront:S#s#1 intern:VP adjnodes:S_VP substnodes:nil DSub:01 SSub:01 DRIO:01 SRIO:01 DRole:Root SRole:Root
3 what WP 6 gave VBD t98 t600 2——cat:WP dsubcat:nil ssubcat:nil comp:n root:NP lfront:nil rfront:nil intern:nil adjnodes:NP substnodes:nil DSub:nil SSub:nil DRIO: SRIO: DRole:1 SRole:0
4 the DT 5 company NN t1 t3 1——cat:D dsubcat:nil ssubcat:nil comp:n root:D lfront:nil rfront:nil intern:nil modif:NP dir:LEFT adjnodes:NP substnodes:nil DSub:nil SSub:nil DRIO: SRIO: DRole:Adj SRole:Adj
5 company NN 6 gave VBD t3 t600 3——cat:N dsubcat:nil ssubcat:nil pred:- comp:n root:NP lfront:nil rfront:nil intern:nil adjnodes:NP substnodes:nil DSub:nil SSub:nil DRIO: SRIO: DRole:0 SRole:1
6 gave VBD 2 know VBP t600 t28 7——cat:V dsubcat:NP0_NP1_NP2(P) ssubcat:NP0_NP1 wh:1 voice:act comp:n datshift:+ root:S lfront:NP#s#1_NP#s#0 rfront:NP#s#2 intern:-NONE-_NP_S_VP adjnodes:NP_S_VP substnodes:nil DSub:012 SSub:01 DRIO:102 SRIO:01 DRole:1 SRole:1
7 the DT 8 president NN t1 t3 1——cat:D dsubcat:nil ssubcat:nil comp:n root:D lfront:nil rfront:nil intern:nil modif:NP dir:LEFT adjnodes:NP substnodes:nil DSub:nil SSub:nil DRIO: SRIO: DRole:Adj SRole:Adj
8 president NN 6 gave VBD t3 t600 8——cat:N dsubcat:nil ssubcat:nil pred:- comp:n root:NP lfront:nil rfront:nil intern:nil adjnodes:NP substnodes:nil DSub:nil SSub:nil DRIO: SRIO: DRole:2 SRole:E
9 . _PERIOD_ 2 know VBP t26 t28 9——cat:. dsubcat:nil ssubcat:nil comp:n root:. lfront:nil rfront:nil intern:nil modif:S dir:RIGHT adjnodes:S substnodes:nil DSub:nil SSub:nil DRIO: SRIO: DRole:Adj SRole:Adj
# lp=-328.112855
...EOS...//...EOS...//...EOS...
...EOS...//...EOS...//...EOS...
...NEW...//...NEW...//...NEW...
...NEW...//...NEW...//...NEW...

## 3 What is a Supertag?

A *supertag* is a tag (i.e., a label or a name) which designates an *elementary tree in a particular TAG grammar*. Since there is a one-to-one relation between supertags and elementary trees in the grammar we will often use these terms interchangeably. Note that the use of a set of supertags (seen as labels) always presupposed a particular TAG grammar, they have no meaning on their own. For more information on supertags, see [Bangalore and Joshi, 1999] (`http://aclweb.org/anthology-new/J/J99/J99-2004.pdf`). This release of the MICA parser uses a grammar extracted by John Chen from the Penn Treebank: [Chen, 2001].

## 4 Attributes

This section lists the attributes that are output as part of MICA if you choose the `-R` option.

### 4.1 Attributes Describing an Elementary Tree

Several attributes describe the elementary TAG tree (i.e., supertag — recall that trees and supertags are in a one-to-one correspondence) in isolation, irrespective of how it is used in a derivation (i.e., in a parse of a particular sentence). These attributes are associated with the tree in the underlying TAG grammar. The attributes in this subsection simply describe this elementary tree.

1. root: the root node of a tree. The node is represented simply by its PTB label. *Note: this is the root of the elementary tree used for the dependent node being described in the output line, NOT for the derived dependency tree!*
2. lfront: ordered list of frontier nodes to the left of main lexical anchor. The nodes are represented using the syntax *cat#type(#role)*, where *cat* is the PTB category of the node, *type* is *s* for substitution node, *c* for co-anchor, or *f* for footnode, and the optional *role* is the deep-syntactic role. Substitution nodes are typically arguments of the lexical head.
3. rfront: ordered list of frontier nodes to the right of main lexical anchor. The nodes are represented using the same syntax as lfront.
4. intern: list of internal nodes of a tree (at which adjunction can take place), i.e. nodes other than root and frontier nodes. Ordering of this list not entirely clear (depth-first?). The nodes are represented simply by their PTB label.
5. adjnodes: a list of node labels which are available for adjunction (i.e., adjunction to this elementary tree).
6. substnodes: this is always NIL. This is a bug.
7. modif: category modified by the tree of this node, if it is an auxiliary tree which is adjoined to another node.
8. dir: direction of modification (as seen from the modifiee!).
9. coanc: a list of co-anchors of this tree, typically prepositions and particles. (Note: it may the case that only one co-anchor appears even if there are several. This is a bug.)

## 4.2 Attributes Interpreting an Elementary Tree

These attributes are also about the tree/supertag in isolation, but they don't merely describe the shape of the elementary tree referenced by a supertag, but interpret them in terms of more abstract linguistic categories.

- voice: this is grammatical voice. Possible values:
  - act – active voice.
  - pas – passive voice without *by*-agent.
  - pas-by – passive voice with *by*-agent. Note that if a *by*-agent is detected, the preposition is a co-anchor. Of course, sometimes a *by*-PP is falsely analyzed (as being an agent when it is not,or not being an agent when it is).

  This feature is only printed for verbs.
- dsubcat: this is the subcategorization frame of the word after undoing passivization and dative shifts. The arguments are listed as numbers with the following meaning:
  - 0 – the deep subject.
  - 0 – the deep direct object.
  - 0 – the deep indirect object.
- ssubcat: this is the surface subcategorization frame. It is different from dsubcat only of voice is not active and dative shift has not occurred. The assignment of surface roles is not always convincing, the deep roles are more reliable. This is a bug.
- DRIO: this is the same list as dsubcat, but in the order in which the arguments occur in the surface string.
- SRIO: this is the same list as ssubcat, but in the order in which the arguments occur in the surface string. This should normally be the same as ssubcat.
- comp: does this verb have a complementizer? Used only for verbs.
- predaux: is this tree a predicative auxiliary tree? Usually, auxiliary trees (those that adjoin) are associated with linguistic modification, but sometimes a matrix clause verb gets an auxiliary tree. Such trees are called "predicative auxiliary".
- particle: if there is a particle, has it shifted? Possible values:
  - + – particle shift, i.e., NP precedes PRT.
  - - – no particle shift, i.e., PRT precedes NP.
  - na – no NP in VP.
  - o – some other case applies.

This feature only appears for particle verbs. When this feature is absent, a fifth value can be assumed (say nopv).

- datshift: if there are two objects and dative shift has taken place (the indirect object precedes the direct object and has no strongly governed preposition), then this attribute is present with value +.

  This feature only appears for active dative verbs. When this feature is absent, a fifth value can be assumed (say nodat).

- wh: this shows if one of the arguments has undergone *wh*-movement. The value is an integer which shows the deep syntactic argument which has undergone *wh*-movement.

- rel: this shows if the verb heads a relative clause. The value is + if yes. This feature only appears if its value is +.

### 4.3 Attributes About Derivations

While the attributes described above are simply associated with a particular supertag, no matter how it is used in a derivation, the attributes described in this section interact with other trees in a derivation.

- drole: this shows the deep syntactic role of this word in this particular derivation. Possible values are as follows:
  - 0, 1, 2: deep syntactic argument (deep subject, deep direct object, deep indirect object).
  - CoHead: a co-anchor. This is used for closely governed prepositions, including the *by* of *by*-agents.
  - Adj: this is an adjunct.
  - X: the role cannot be determined. This occurs in relative clauses with *that* . This is a bug.
  - E: there is an error in determining the deep syntactic role. This is a bug.
  - Root: this is the root of the derived dependency tree. There should be only one word per sentence with this Drole.

- srole: this shows the surface syntactic role of this word in this particular derivation. Possible values are as above for drole. The notion of surface role is not well defined and should be used with some caution.

## References

[Bangalore et al., 2009] Bangalore, S., Boullier, P., Nasr, A., Rambow, O., and Sagot, B. (2009). MICA: A probabilistic dependency parser based on tree insertion grammars. In *NAACL HLT 2009 (Short Papers).*

[Bangalore and Joshi, 1999] Bangalore, S. and Joshi, A. (1999). Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–266.

[Chen, 2001] Chen, J. (2001). *Towards Efficient Statistical Parsing Using Lexicalized Grammatical Information*. PhD thesis, University of Delaware.