Sanchay Syntactic Annotation Interface

Anil Kumar Singh

<anil@research.iiit.ac.in>

Language Technologies Research Centre

International Institute of Information Technology

Hyderabad, India

Introduction

- Sanchay: A collection of APIs and tools for NLP

- Includes GUI based interfaces for annotation
  - Like Syntactic Annotation Interface

Syntactic Annotation Interface

- Can view or edit:
  - POS tagging
  - Chunking
  - Feature structures of words or chunks
    - Morphological information
    - Kaaraka or dependency relationships
- Many other things
  - Anything in the form of a tree
  - (Optional) Where a node can have feature structures

Overview

- System requirements and Installation

- Working modes

  - Stand-alone or task mode

- Setting up the tasks

- Starting the interface

- Viewing annotation

- Editing annotation

- Comparing annotation

System Requirements

- JDK-1.5 (or JRE-1.5) installed on your system

- Preferably >1 GHz processor and >256 MB RAM

- You may like to specify the memory allocated for JVM (see later), if you have more RAM

- Tested on Linux and Windows

Installation

- Untar the Sanchay.tgz file somewhere on your system
    - Or unzip the Sanchay.zip file
- There should be a **Sanchay** directory
- That's all for installation!

Working Modes

- You can work in two modes
  - Stand-alone mode
    - Directly open a the corpus file (Browse)
    - Some assumptions: e.g., specific separators
  - Task mode
    - Select the task group
    - Then select the task

Which Mode is Better?

- Stand-alone is better if
  - You don't know how to set up tasks
  - You don't want to set up tasks
  - You only need to use the interface occasionally
- Task mode is better if
  - You use the interface quite often or on many files
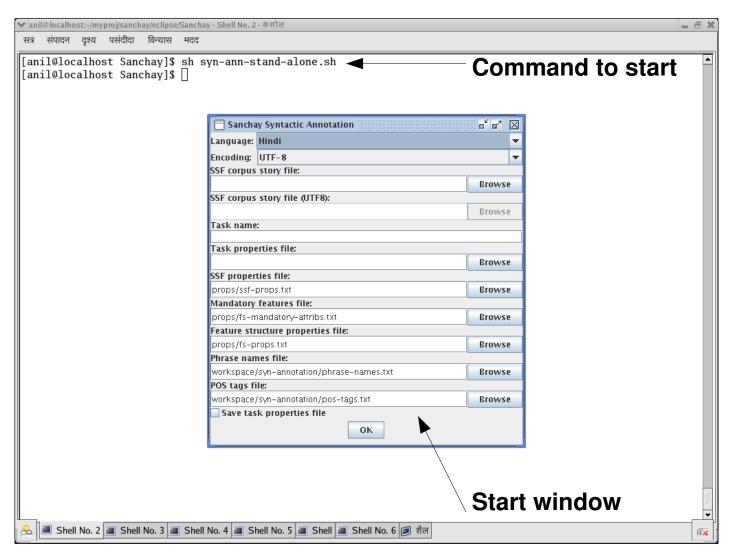 - You need to customize certain things

Working in Stand-alone Mode-I

- **On Linux**
  1. On the console, go to the Sanchay directory
  2. Run the shell script
     sh syn-ann-stand-alone.sh

- **On Windows**
  1. Go to the Sanchay directory
  2. Double click on
     syn-ann-stand-alone.bat
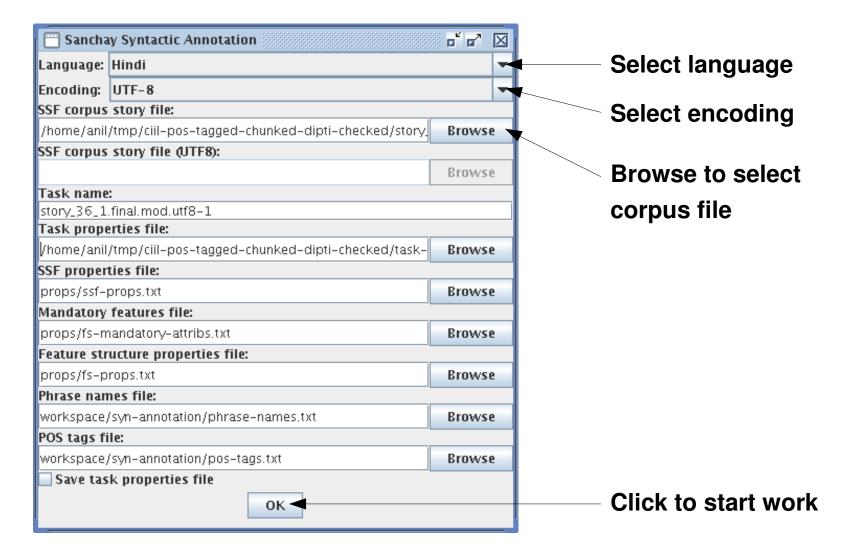   Or **syn-ann-stand-alone** (if DOS extensions are not displayed)

Working in Stand-alone Mode-II

- You will see the task properties window

- Select the language and encoding

- Select the corpus file to be annotated

    - By clicking on the first (top) Browse button

- Normally, you don't have to fill in anything else

- Click on the OK button to start work

- You will see the work window where one sentence from the task data file will be displayed

Working in Stand-alone Mode-III
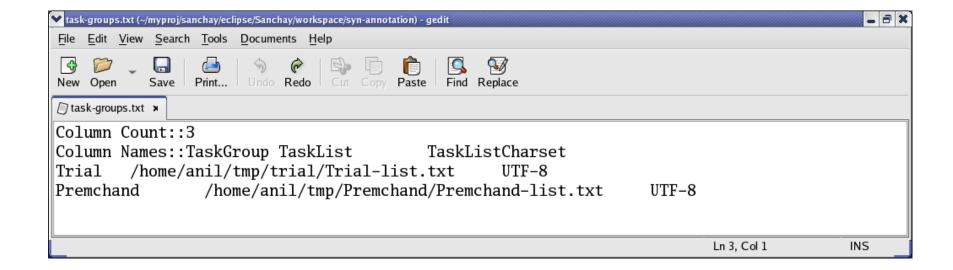
The Main Window (Stand-alone)

Task Setup-I

- Annotation performed on 'tasks'

- Details about a task in a properties file

- Task data located inside:

  - **Sanchay**/**workspace**/**syn-annotation** (default)

  - Or any other directory on your system

- You can form tasks groups

- (Preferably) One directory for each task group

Task Setup-II

- E.g., an **nlpai-tasks** directory for NLPAI ML contest tasks
- Each 'task' represents a part of the (to be) annotated corpus
  - A 'story' or a part of the 'story'
- A list of task groups is kept in a file **task-groups.txt** in **syn-annotation** directory
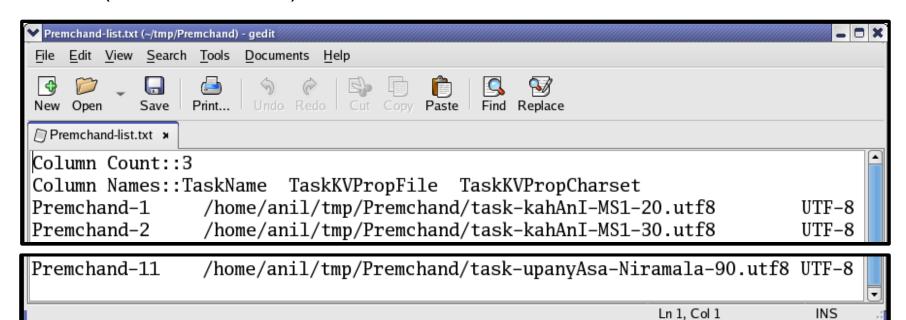- Each entry in this file refers to one task list

Task Setup-III

- **task-groups.txt** has tab separated fields
  - Task group name, task list file path and charset of this file (default UTF-8)

Task Setup-IV

- Each task list file, say **Premchand-list.txt** has tab separated fields
  - Task name, task properties file path and charset of this file (default UTF-8)
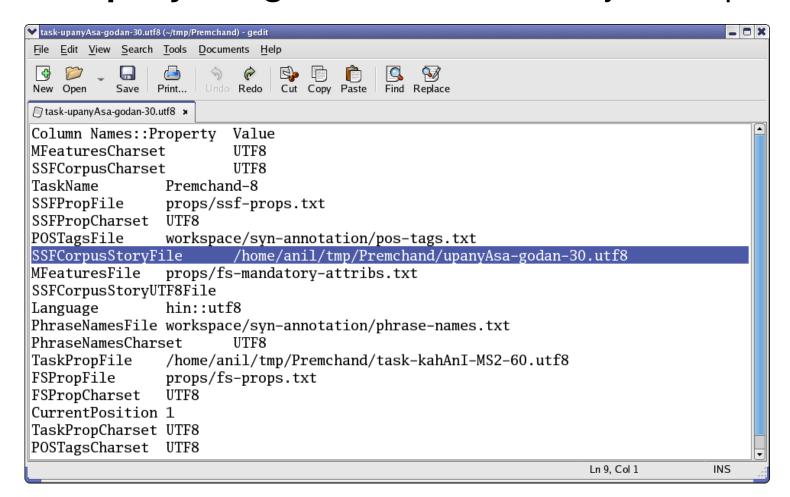
Task Properties Files-I

- Each task has a properties file with key-value pairs

- Three main keys for setting up new tasks

  - TaskName

  - SSFCorpusStoryFile

  - TaskPropFile

  - Language: language & encoding code, separated by '::'

    - Hindi (UTF-8) will be specified as hin::utf8

  - Language and encoding codes listed in:

    - Sanchay/props/languages.txt

    - Sanchay/props/encodings.txt

Task Properties Files-II

- E.g., for **Premchand-8**
  - □ There is an entry in **Premchand-list.txt**
  - □ There is a task properties file, say, **task-upanyAsa-godan-30.utf8**
    - Note the prefix **'task-'**
  - □ There is the corpus 'story' part, say, **upanyAsa-godan-30.utf8**
    - The actual corpus data file

Task Properties Files-III

- **task-upanyAsa-godan-30.utf8** has key-value pairs

Notes about Encoding or Font

- The default encoding is UTF-8
- Sorry, but ISCII won't work currently
  - Let's hope it will soon
- You might need to convert corpus files to UTF-8 or WX etc.

Corpus Data Format Disclaimer

- The interface only accepts data in correct SSF format

  - Actually, raw data (simple text) is also accepted, but is converted into SSF (see the next slide)

- SSF is Shakti Standard Format

  - A CFG (Context Free Grammar) tree like format with the addition that every node can have feature structures for carrying morphological and dependency relation information, among other things

WARNING - Using Raw Data

- The task data files can also be in raw form
  - **ONE SENTENCE PER LINE**
- If start work on a task with raw data
  - You get a message about raw data
  - You are asked to make sure each sentence is on a separate line
  - You are presented with a text editor for this
- A preprocessing program is run on the raw data to break up the sentence and to do tokenization, but you may have to check for errors

SSF

Shakti Standard Format

```
                                        chunk-name
                                          /
                                         /
1          ((                      NP
1.1        children                NNS     <af=child,n,m,p,3,0,,>
           ))   \                   |              |   | | |  | | |
                |                   |              |   | | |  | | \
                |                   |             root | |  |pers \
              lex                  pos                 | |  |   case
                                               cat | number
                                               gender

2          ((                      VG
2.1        are                     VBP     <af=be,v,m,p,3,0,,>
2.2        watching                VBG     <af=watch,v,m,s,3,0,, /aspect='PROG'>
           ))
3          ((                      NP
3.1        some                    DT      <af=some,D,m,s,3,0,,>|<af=some,det,m,s,3,0,,>
3.2        programmes              NNS     <af=programme,n,m,p,3,0,,>
           ))

4          ((                      PP
4.1        on                      IN      <af=on,p,m,s,3,0,,>|<af=on,n,m,s,3,0,,>|
4.1.1      ((                      NP
4.1.2      television              NN      <af=television,n,m,s,3,0,,>
           ))
           ))

5          ((                      PP
5.1        in                      IN      <af=in,p,m,s,3,0,,>|<af=in,D,m,s,3,0,,>
5.2        ((                      NP
5.2.1      the                     DT      <af=the,det,m,s,3,0,,>
5.2.2      house                   NN      <af=house,n,m,s,3,0,,>|<af=house,v,m,s,3,0,,>
           ))
           ))
```
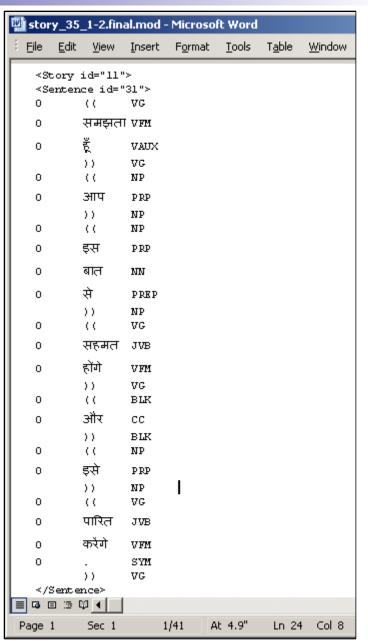
Corpus Data File-I

- Everything between **\<Story>\</Story>** tags
  - This tag is optional now

- Story can have an id: **\<Story id="1">**

- A sentence between **\<Sentence>\</Sentence>** tags

- Sentences must have an id: \<Sentence id="1">

Corpus Data File-II

- Four fields in SSF format

  1. The node id (in way deprecated since the SSF API works on the basis of brackets)
     - These ids are usually automatically allocated
     - If not, some number should be there (say, 0)

  2. Starting ("((") or ending ("))") brackets for chunks or the word for a lexical item

  3. The chunk name for a chunk or the POS tag for a lexical item

  4. The feature structure for the node (chunk or lexical item)

Corpus Data File-III
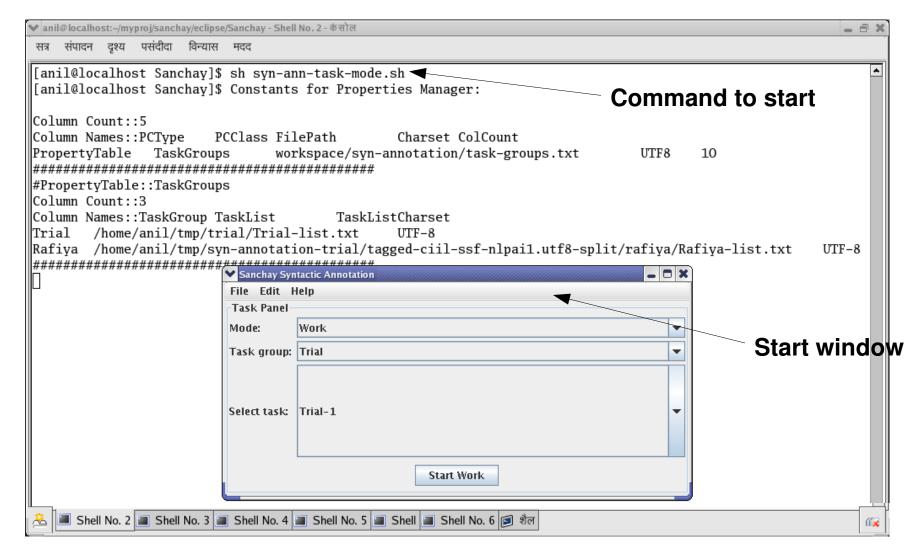
- A sample data file
  - **story_35_1-2.final.mod.utf8**

```
story_35_1-2.final.mod - Microsoft Word
File   Edit   View   Insert   Format   Tools   Table   Window

<Story id="11">
<Sentence id="31">
0        ((        VG
0        समझता     VFM
0        हूँ        VAUX
         ))        VG
0        ((        NP
0        आप        PRP
         ))        NP
0        ((        NP
0        इस        PRP
0        बात       NN
0        से        PREP
         ))        NP
0        ((        VG
0        सहमत      JVB
0        होंगे      VFM
         ))        VG
0        ((        BLK
0        और        CC
         ))        BLK
0        ((        NP
0        इसे       PRP
         ))        NP
0        ((        VG
0        पारित     JVB
0        करेंगे     VFM
0        .         SYM
         ))        VG
</Sentence>
```
```
Page 1      Sec 1          1/41      At 4.9"      Ln 24    Col 8
```

Working in Task Mode-I

- On Linux
    1. On the console, go to the Sanchay directory
    2. Run the shell script

        sh syn-ann-task-mode.sh

- On Windows
    1. Go to the Sanchay directory
    2. Double click on

        syn-ann-task-mode.bat

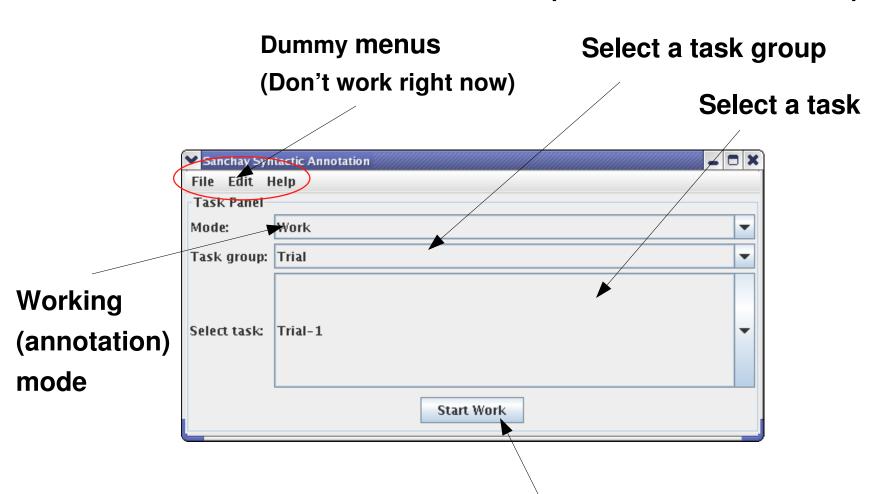   Or **syn-ann-task-mode** (if DOS extensions are not displayed)

Working in Task Mode-II

- You will see the task window

- You can select the task group and the task that you want to view/edit

- Select the task, click on the **Start Work** button

- You will see the work window where one sentence from the task data file will be displayed

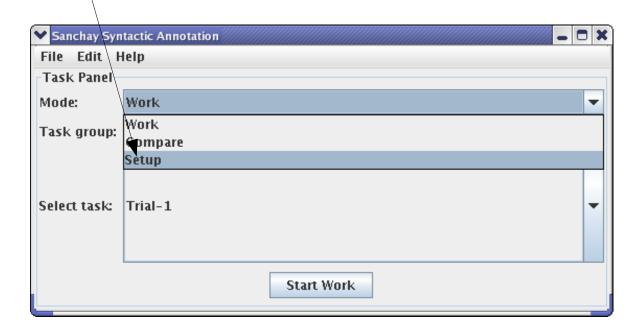- You can now perform/edit annotation

Working in Task Mode-III

The Main Window (Task Mode)

Dummy menus

(Don't work right now)

Select a task group

Select a task

Working

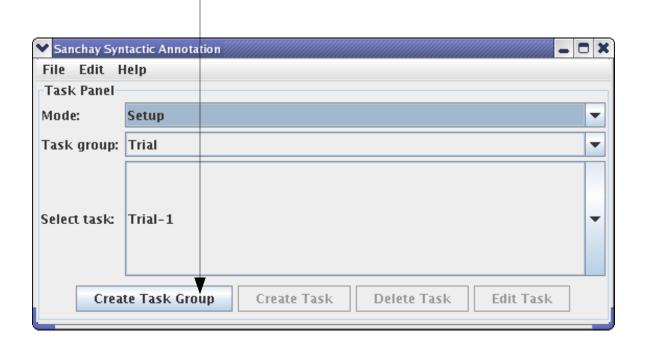**(annotation)**

mode

Click to start working

Creating Task Groups-I

- There is a program to easily setup tasks
- Select Setup mode from start window

•Creating Task Groups-II

■ Click on **Create Task Group** button

•Creating Task Groups-III

■ Fill in the task group details

Task group name

Task properties template (an example properties file which will be copied with some fields being changed)

Directory where task properties files will be stored

Add task data (corpus) files

Currently not available
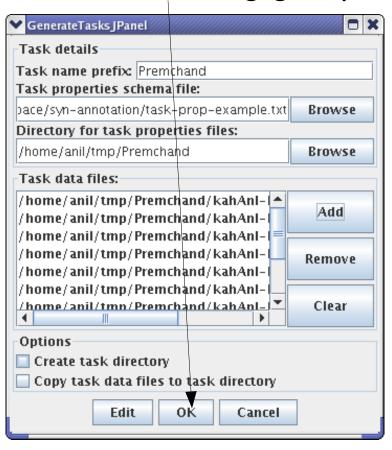
•Creating Task Groups-IV
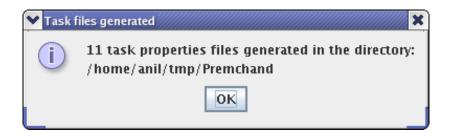
■ Adding corpus data files after clicking on **Add**

•Creating Task Groups-V

■ Finish creating groups
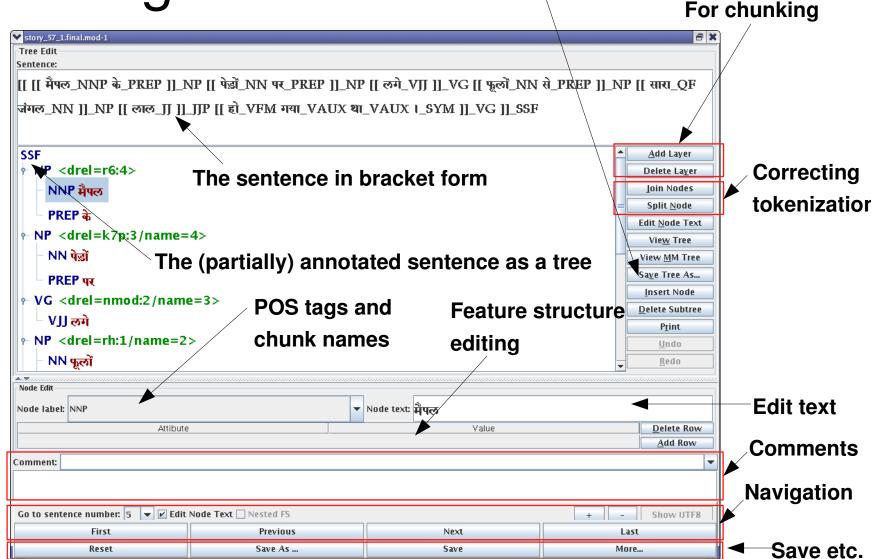
Working Directory Disclaimer

- Currently Sanchay will work only if you start it from the **Sanchay** directory

- Thus, you can't create shortcuts somewhere else unless you take care that the current directory gets changed to **Sanchay** before the above script is run
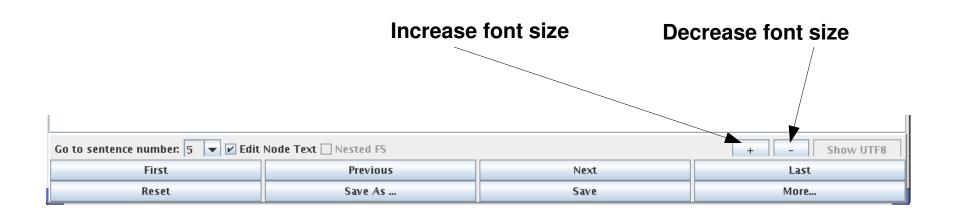
Viewing Annotation

Saving the sentence tree separately

For chunking

Correcting tokenization

The sentence in bracket form

**The (partially) annotated sentence as a tree**

POS tags and chunk names

Feature structure editing

Edit text

Comments

Navigation

Save etc.

story_57_1.final.mod-1

Tree Edit

Sentence:

[[[[मैपल_NNP के_PREP]]_NP [[पेड़ों_NN पर_PREP]]_NP [[लगे_VJJ]]_VG [[फूलों_NN से_PREP]]_NP [[सारा_QF जंगल_NN]]_NP [[लाल_JJ]]_JJP [[हो_VFM गया_VAUX था_VAUX ।_SYM]]_VG]]_SSF

SSF
- NP <drel=r6:4>
  - NNP मैपल
  - PREP के
- NP <drel=k7p:3/name=4>
  - NN पेड़ों
  - PREP पर
- VG <drel=nmod:2/name=3>
  - VJJ लगे
- NP <drel=rh:1/name=2>
  - NN फूलों

Add Layer
Delete Layer
Join Nodes
Split Node
Edit Node Text
View Tree
View MM Tree
Save Tree As...
Insert Node
Delete Subtree
Print
Undo
Redo

Node Edit

Node label: NNP   Node text: मैपल

Attribute	Value	
		Delete Row
		Add Row

Comment:

Go to sentence number: 5 ☑ Edit Node Text ☐ Nested FS   +   −   Show UTF8

First	Previous	Next	Last
Reset	Save As ...	Save	More...

Increase/Decrease Font Size

- You can change the font size by clicking on these buttons

Increase font size   **Decrease font size**

Go to sentence number: 5 ▾ ☑ Edit Node Text ☐ Nested FS			+	–	Show UTF8
First	Previous	Next		Last	
Reset	Save As ...	Save		More...	

Navigation

- At a time only one sentence is displayed in the work widow

- Using the buttons in the navigation bar, you can **go to** any sentence in the task
  - First, Previous, Next, Last
  - Or any specific sentence by selecting its number from the list (combo box)

POS Tagging

- Select (say, by clicking) a lexical item (word) in the tree area

- You will see the list of possible POS tags in the **Node label** list (combo box)

- From the **Node label** list, select a tag

- The new POS tag for the word will be displayed in the tree area

Chunks vs. Phrases

- The interface allows arbitrary nesting
  - PP inside an NP, which is itself inside an NP
- But by chunking we mean only one level
  - A chunk can be NP, VG etc. and can only contain lexical items inside it, not chunks
  - No recursion
  - NP -> The book on the table
    - But not
      - NP -> NP PP **or** PP -> Prep NP

Phrases or Recursion Not Allowed

- It is assumed that only chunks will be annotated
- Everything (for the time being) will be in terms of
  - Words, POS tags, chunks, feature structures

Forming Chunks

- You want to chunk "*The red book*"

- Select these words together in order

- You can do this by pressing the Ctrl or Shift and clicking on the words

- Click on **Add Layer** button

- A new node will be created and the words you selected will become child nodes of this node

Naming Chunks

- Select the chunk node (say, NP)

- You will see the list of possible chunk names in the **Node label** list (combo box)

- From the **Node label** list, select a chunk name

- The new chunk name for the chunk will be displayed in the tree area

Removing Chunks

- Now suppose a chunk is wrongly formed

- Select the chunk node (say, NP)

- Click on **Delete Layer** button

- The chunk node will be removed

- The words will move up in the tree

Forming and Removing

- For any rearrangement of words into chunks, you will have to use combinations of **form** and **remove** chunk operations

- Note that words to be combined into a chunk should be on the same level

- You may have to bring them on the same level by removing those chunks they are currently parts, if any

Combining Chunks

- You want to combine two chunks

- Remove the first chunk as explained above

- Remove the second chunk

- Combine the words of the two chunks into one chunk

Splitting Chunks

- You want to split one chunk into two

- Remove the chunk

- Select some words and form a new chunk

- Select the rest of the words into another chunk

Feature Structure Annotation-I

- Beyond tagging and chunking

- Every node (word or chunk) in the tree can have a feature structure
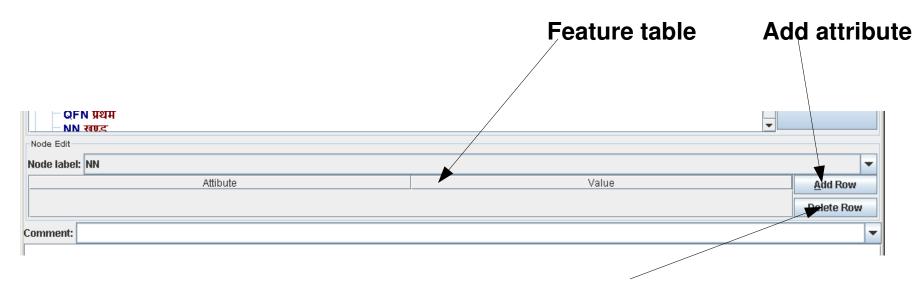
Feature Structure

- A feature structure has attribute value pairs

- Some attributes may be mandatory

- The attribute value can itself be a feature structure

  - Usually it will just be a string

  - The current interface assumes only string values

Feature Structure Annotation-II

- When you select a node in the tree, you will see a feature table showing the attributes and values
  - May be empty

Feature table   **Add attribute**

Delete attribute

Adding Attributes

- Select the node for which you want to add an attribute

- You will see the attribute table below the **Node label**

- Click on the **Add Row** button

- There will be a new row representing a new attribute in the feature table
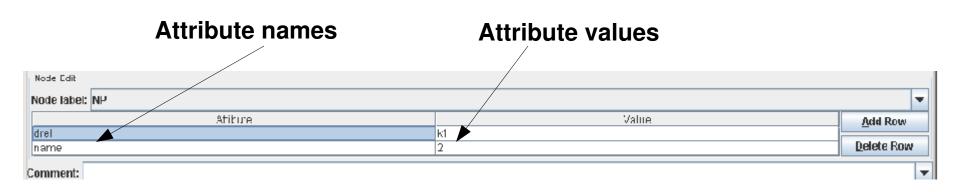
Naming Attributes

- In the feature table, select the attribute (which may have no name if it was just created)
  - By clicking on the table cell under the **Attribute** column
- Type the new name and press enter
- When you click on a node other than the currently selected one in the tree, the new attribute will appear near the node

Setting/Changing Attribute Values

- In the feature table, select the attribute value (which may be empty)

  - By clicking on the table cell under the **Value** column

- Type the new value and press enter

- When you click on a node other than the currently selected one in the tree, the new attribute value will appear near the node

Attribute-Value Pairs

Attribute names

Attribute values

Node Edit

Node label: NP

Atibure	Value	
drel	k1	**Add Row**
name	2	**Delete Row**

Comment:

Displayed feature structure

SSF

NP <drel=k1/name=2>

NNP निराला

PREP की

NP

Deleting Attributes

- In the feature table, select the row for the attribute you want to delete

  - By clicking any cell in the row

- Click on the **Delete Row** button in the feature table

- When you click on a node other than the currently selected one in the tree, you can see that the attribute has been deleted

Syntactic Annotation

- By syntactic annotation we mean marking up dependency relations
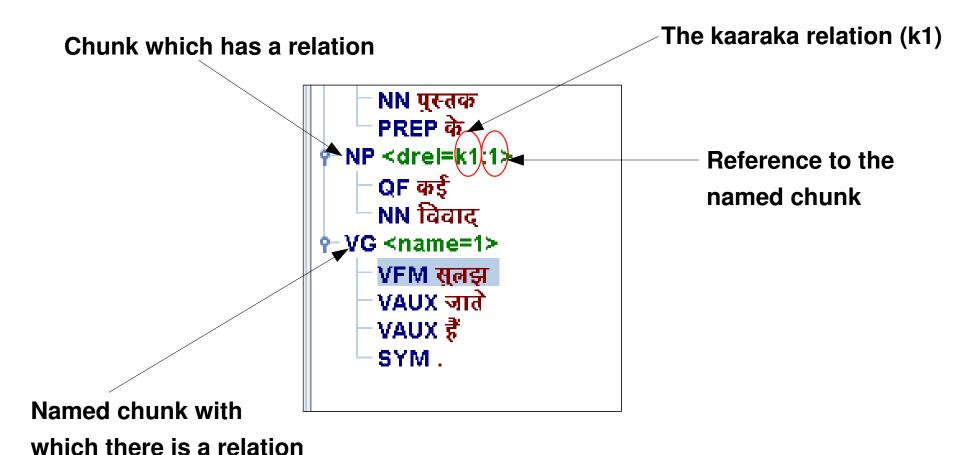 - Say, kaaraka relations

Kaaraka Relations-I

- The main thing is
  - **Which node has what relation with which other node**
- The node which 'has the relation' is given an attribute called **drel** whose value (say, **k1** or kartaa) will be the kaaraka relation
- The with which it has the relation is given an attribute called **name**
- The names should have unique values (across a sentence)
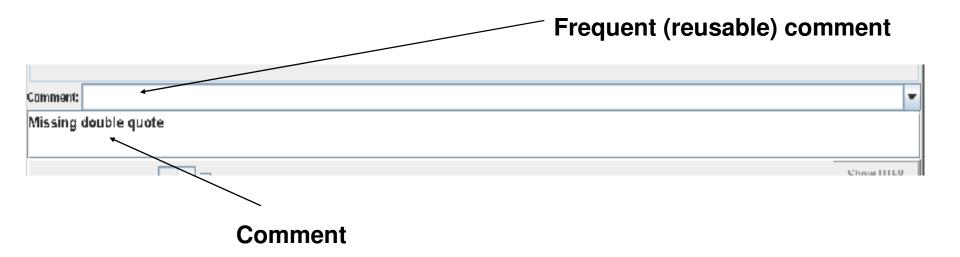
Kaaraka Relations-II

- Name uniquely identifies a chunk with which other chunks are related

- This name is referred to in the value of the **drel** attribute of the chunk which has a relation with the named chunk

- This is done by adding the name to the value of the drel attribute, separated by a colon

Kaaraka Relations-III

Chunk which has a relation

The kaaraka relation (k1)

```
        NN पुस्तक
        PREP के
NP <drel=k1:1>
        QF कई
        NN विवाद
VG <name=1>
        VFM सुलझा
        VAUX जाते
        VAUX हैं
        SYM .
```

Reference to the named chunk

Named chunk with which there is a relation

Comments

- For any sentence, if there is any (say, linguistic) issue which needs to be noted down, you can add a comment
  - Or a typing/grammatical mistake etc.

Frequent (reusable) comment

Comment: [_____] ▼

Missing double quote

Comment

Reusable Comments

- In the combo box above the comment text box, you can type comments which are needed frequently
  - So that you don't have to type them again and again
- Once you have entered them in the combo box, you can just select them later from the list (combo box) and they will appear in the comment text box

Saving

- There are three buttons for 'saving'

- **Save** actually saves the complete task

  - The one you will mostly use

- **Save Tree As** saves one sentence in (SSF format, text file) wherever you want

- **Save As** saves the complete task in another format wherever you want

  - XML currently not implemented

Reset and Clear

- **Reset** will discard recent work and display the sentence read from the saved task

- **Clear** will clear tagging and chunking and will present the current sentence as a sequence of words

- **Reset All** will discard all the work you did after pressing **Save** and read the whole task again from the task file

Caution

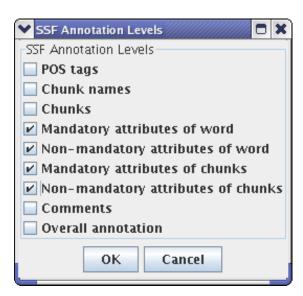- Use **Reset**, **Clear** and **Reset All** buttons carefully

More...

- You can see some more buttons by clicking on **More**
- **Clear All** for clearing some annotation level (see next slide) from all sentences in the task
- **Clear** for clearing some annotation level (see next slide) from the current sentence
- **Reset All** for reloading all sentences in the task
- **Join Sentence** and **Split Sentence**

Clear Annotation Levels

- Clicking on **Clear** and **Clear All** will show you a dialog to select the annotation levels which you want to clear
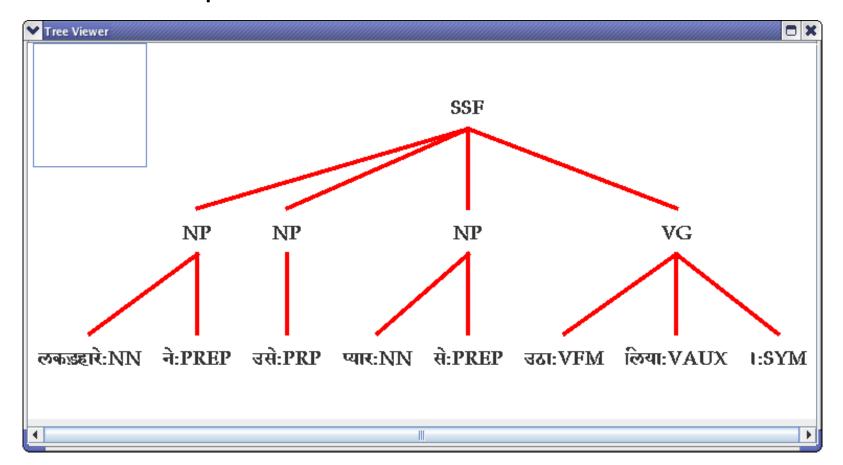
- A combination of specific levels can be cleared

Bracket Form

- A tree based representation of sentence chunking or parsing

- (Possibly nested) brackets mark the chunks or phrases
  - For us, only chunks

- Allows you to see the chunking horizontally, rather than vertically (as in a tree)
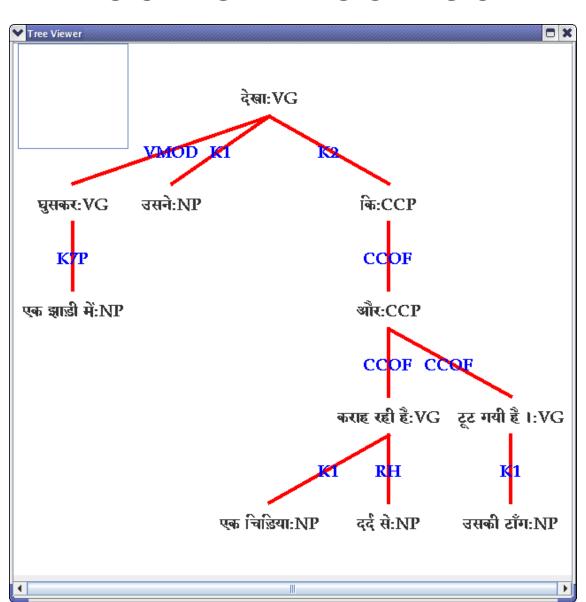
•Tree Viewer: Phrase Structure

- To view the phrase structure tree, click on **View Tree**
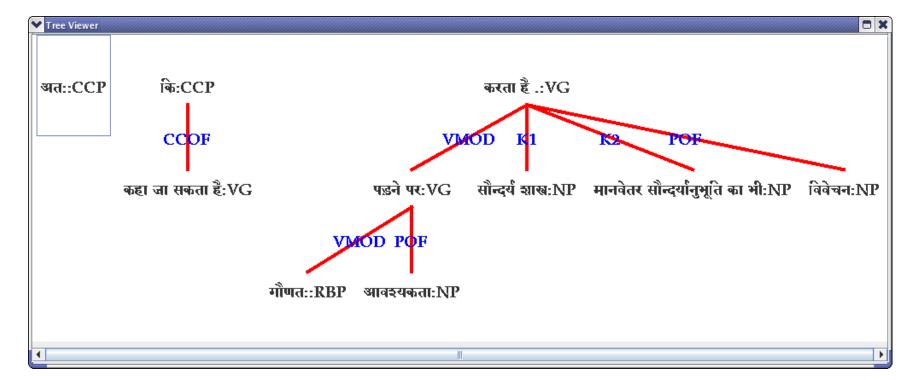
•Tree Viewer: Modifier-Modified Tree

- To view the modifier-modified tree, click on **View MM Tree**

•Modifier-Modified Tree

- Some of the mistakes in annotation might become apparent just by looking at the MM-tree
  - □ Disconnected sub-trees

•Other Kinds of Annotation

- Use the same interface for other kinds of annotation
  - Where the data can be represented as a tree (optionally) with nodes having feature structures

•Example: Named Entities

- Open the file
  - workspace/syn-annotation/phrase-names.txt
- Change the list
  - So that phrase names are named entity types
- You may or may not use POS tags and feature structures

•Other Examples

- Similarly, you can adopt the interface for marking up
  - Multi-word expressions
  - Temporal expressions
  - Dependency relations
    - Other than kaaraka relations (default)
  - Etc.

Increasing JVM Memory-I

- The memory allocated by default to JVM (heap size) may not be enough, given the RAM on your system

- Suppose you have 512MB RAM, you might want to allocate more memory to JVM to make Sanchay run faster

- Change the **run-syntactic-annotation.sh** or **run-syntactic-annotation.bat** script

Increasing JVM Memory-II

- Edit the shell/batch files used for starting the interface
- Use the –Xms option to specify the initial heap size
- Use the –Xmx option to specify the maximum heap size
- Example
    - java –Xms128m –Xmx256m …

Concluding Disclaimer

- **Sanchay** is at present continuously evolving

- Parts of it are being used for practical work, but the design has yet to stabilize

- There can be radical changes

- Fortunately, the data created by you will be usable even in future, as it is in a standard format

- Any comments/bugs
  - Mail at

    anil@research.iiit.ac.in