CS6770 - Conceptual Dependency Theory - 1a

Pranali Yawalkar - CS11B046

Tanmay Dhote - CS11B051

pranali.yawalkar@gmail.com

tanmaydhote310.3@gmail.com

III Year Undergraduate students
IIT Madras

April 28, 2014

Abstract

Conceptual dependency theory is a model of natural language understanding used in artificial intelligence systems. Roger Schank at Stanford University introduced the model in 1969, in the early days of artificial intelligence. This model was extensively used by Schank's students at Yale University such as Robert Wilensky, Wendy Lehnert, and Janet Kolodner. Schank developed the model to represent knowledge for natural language input into computers. Partly influenced by the work of Sydney Lamb, his goal was to make the meaning independent of the words used in the input, i.e. two sentences identical in meaning, would have a single representation. The system was also intended to draw logical inferences, The model uses the following basic representational tokens - real world objects and their attributes, real world actions and their attributes, times and locations.

I. Problem Definition

Design and construct a lexicon of 800-1000 English words. Include different senses of the words, and the semantic constraints on the different role fillers.

II. Domain of consideration

Indian General Elections 2014 - All chosen words in our dictionary have some relevance to the current political scenario in our country A list of words of various types has been created in different files as follows -

- ADJECTIVE_NAMES
- ADVERB_NAMES
- ATRANS
- ATTEND
- DO_SOMETHING
- EXPEL
- GRASP
- INGEST
- MBUILD
- MTRANS
- OBJECT_NAMES
- PLACES_NAMES
- POLITICIANS_NAMES
- PRONOUN_NAMES
- PROPEL
- PTRANS
- SPEAK
- STATE

The description of these words is as follows -

- ADJECTIVE_NAMES - It contains a list of adjectives known as picture aiders in CD theory. A list of 80 adjectives has been put in the list. Eg - political, massive
- ADVERB_NAMES - It contains a list of adverbs known as action aiders in CD theory. A list of 33 adjectives has been put in the list. Eg - Selfishly, Badly

- ATRANS - It contains a list of verbs known that correspond to ATRANS verbs in the CD theory. It also contains the result of the actions. A list of 23 distinct verbs has been put in the list along with their different tenses. Eg - bestow, present
- ATTEND - It contains a list of verbs known that correspond to ATTEND verbs in the CD theory. It also contains the result of the actions. A list of 16 distinct verbs has been put in the list along with their different tenses. Eg - hear, look
- DO_SOMETHING - It contains a list of verbs that correspond to CD verbs that cause a state change in either the subject or the object without actually specifying what is done (hence DO_SOMETHING). The list in the file also contains the result of the actions. A list of 36 distinct verbs has been put in the list along with their different tenses. Eg - kill,hate
- EXPEL - It contains a list of verbs known that correspond to EXPEL verbs in the CD theory. It also contains the result of the actions. A list of 4 distinct verbs has been put in the list along with their different tenses. Eg - bleed, sweat
- GRASP - It contains a list of verbs known that correspond to GRASP verbs in the CD theory. It also contains the result of the actions. A list of 2 distinct verbs has been put in the list along with their different tenses. Eg - grab, hold
- INGEST - It contains a list of verbs known that correspond to INGEST verbs in the CD theory. It also contains the result of the actions. A list of 2 distinct verbs has been put in the list along with their different tenses. Eg - eat, drink
- MBUILD - It contains a list of verbs known that correspond to MBUILD verbs in the CD theory. It also contains the result of the actions. A list of 16 distinct verbs has been put in the list along with their different tenses. Eg - decide, speculate
- MTRANS - It contains a list of verbs known that correspond to MTRANS verbs in the CD theory. It also contains the result of the actions. A list of 31 distinct verbs has been put in the list along with their different tenses. Eg - confess, teach
- OBJECT_NAMES - It contains a list of objects (both real and abstract). A list of 82 distinct objects has been put in the list. Eg - liquor, revenge
- PLACES_NAMES - It contains a list of places (constituencies). A list of 543 distinct constituencies has been put in the list. Eg - Varanasi, Chandni Chowk
- POLITICIANS_NAMES - It contains a list of politicians as well as some common nouns that can act as an actor. A list of 118 such nouns has been put in the list. Eg - NARENDRA_MODI, Son
- PRONOUN_NAMES - It contains a list of pronouns. It also contains the information whether the pronouns correspond to an object or a person and whether it is singular or plural. A list of 55 distinct pronouns has been put in the list. Eg - he, we
- PROPEL - It contains a list of verbs known that correspond to PROPEL verbs in the CD theory. It also contains the result of the actions. A list of 7 distinct verbs has been put in the list along with their different tenses. Eg - shoot, pelt
- PTRANS - It contains a list of verbs known that correspond to PTRANS verbs in the CD theory. It also contains the result of the actions. A list of 31 distinct verbs has been put in the list along with their different tenses. Eg - land, appear

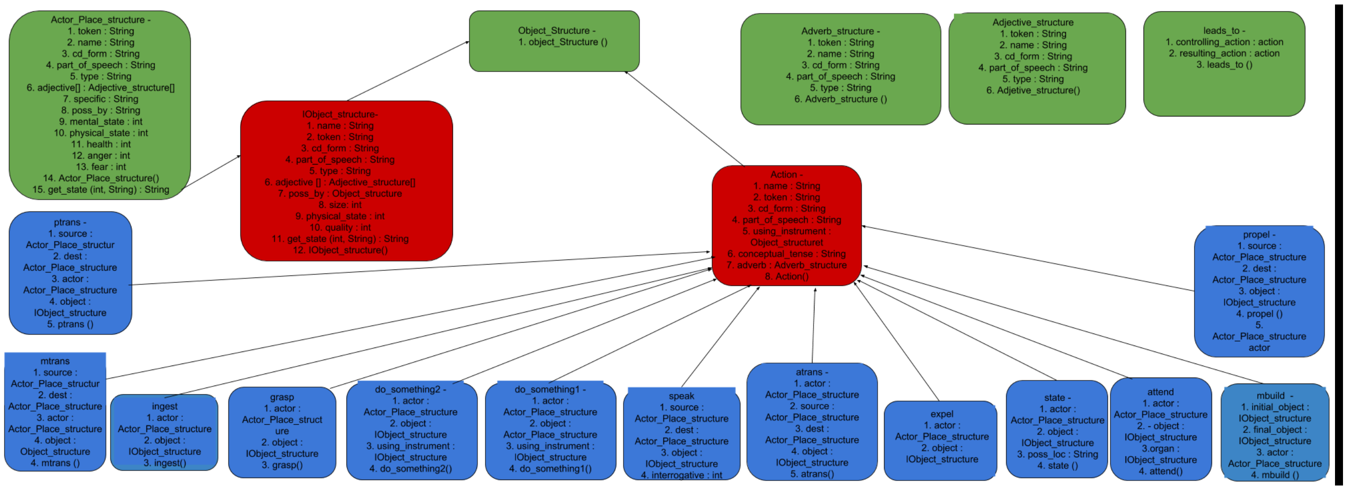
- SPEAK - It contains a list of verbs known that correspond to SPEAK verbs in the CD theory. It also contains the result of the actions. A list of 38 distinct verbs has been put in the list along with their different tenses. Eg - criticise, say
- STATE - It contains a list of verbs known that causes a state change. A list of 18 distinct verbs has been put in the list along with their different tenses. Eg - forget, mourn

III. Generating the lexicon in structure

The JAVA code reads inputs from the above mentioned files into the corresponding data structures for each type of these words. This data structure contains the appropriate CD form representation of the type of the word. It basically gives a template that has to be filled while parsing.
The fields that have been used for determining the various entities are :

- token - unique identifier for that entity
- name - entity name
- cd_form - CD form of that entity
- part_of_speech - part of speech of that entity (eg. noun, verb)
- conceptual_tense :
  - past
  - future
  - continuing
  - Interrogative
  - negative
  - present

The figure shows the class hierarchy of the various classes of words, the fields contained by them, their data type and the relation among the classes. An edge corresponds to inheritance. A -> B means that A is B's derived class. This is based on the "Decorator design pattern" of object oriented design patterns. The entire structure has a maximum of 3 level class heirarchy.



The green nodes are the ones that are never derived from any other class. The red ones are are on level 2 of the heirarchy and the blue nodes are on the level 3 of the hierarchy.
Description of the classes -

- **Object_structure** : A class Object_structure is defined initially. This structure is empty and is built intentionally so that actions can also be treated as objects for certain actions. Eg, John remembered that Mary ate an apple. Here Mary eating an apple is an object for the 'remember' action of John. To enable such structures programmatically, the empty base class has been introduced and the true Object class and Action class inherit from this one.

- **action** : A class action has been defined which inherits from the empty class Object_strucure. The fields present in it are -

  - name
  - token
  - cd_form
  - part_of_speech
  - using_instrument - it can either be the instrumental act or the object used in doing the action
  - conceptual_tense
  - adverb - the action aider for this verb

- **ADJECTIVE_NAMES** : The corresponding java file is Adjective_structure.java. The fields present in it are

  - token
  - name
  - cd_form - (ADJECTIVE VAR1)
  - part_of_speech

- **ADVERB_NAMES** : The corresponding java file is Adverb_structure.java. The fields present in it are
  - token
  - name
  - cd_form - (ADVERB VAR1)
  - part_of_speech

- **ATRANS** - The corresponding java file is atrans.java. This class extends the action class. The additional fields present in it are
  - actor - who does the action
  - source - initial owner
  - destination - final owner
  - object - the object whose ownership is changed
  - cd_form -
    (ATRANS
    (FROM VAR1)
    (TO VAR2)
    (OBJECT VAR3)
    (ACTOR VAR4)
    (CONC_TENSE VAR5)
    (USING VAR6)
    (ADVERB VAR7)
)

- **ATTEND** - The corresponding java file is attend.java. This class extends the action class. The additional fields present in it are
  - actor - who performs the action
  - organ - organ that is used
  - object - the recipient of the action
  - cd-form -
    (ATTEND
    (ACTOR VAR1)
    (ORGAN VAR2)
    (OBJECT VAR3)
    (CONC_TENSE VAR4)
    (USING VAR5)
    (ADVERB VAR6)
)

- **DO_SOMETHING** - The corresponding java file is do_something.java. This class extends the action class. The additional fields present in it are
  - actor - who performs the action
  - object - on whom the action is performed
  - cd-form -
    (DO SOMETHING
    (ACTOR VAR1)
    (OBJECT VAR2)
    (CONC_TENSE VAR3)
    (USING VAR4)
    (ADVERB VAR5)
)

  The actor does something on the object which causes a state change in either of them.We have clearly distinguished between the case when the action is performed on a person and place, and when it is performed on a physical or abstract object. There are 2 do_something class structure provided - do_something1 and do_something2. do_something specifies the object of the action to be of type person or place and do_something2 specifies the object of the action to be of an object type.

- **EXPEL** - The corresponding java file is expel.java. This class extends the action class. The additional fields present in it are
  - actor - who does the action
  - object - the object that is expelled
  - cd-form -
    (EXPEL
    (FROM VAR1)
    (OBJECT VAR2)
    (CONC_TENSE VAR3)
    (USING VAR4)
    (ADVERB VAR5)
)

- **GRASP** - The corresponding java file is grasp.java. This class extends the action class. The additional fields present in it are
  - actor - who does the action
  - object - the object which is grasped
  - cd-form -
    (GRASP
    (ACTOR VAR1)
    (OBJECT VAR2)
    (CONC_TENSE VAR3)
    (USING VAR4)

(ADVERB VAR5)
)

- **INGEST** - The corresponding java file is ingest.java. This class extends the action class. The additional fields present in it are
  - actor - who does the action
  - object - the object on which the action takes place
  - cd - form -
    (INGEST
    (ACTOR VAR1)
    (OBJECT VAR2)
    (CONC_TENSE VAR3)
    (USING VAR4)
    (ADVERB VAR5)
)

- **MBUILD** - The corresponding java file is mbuild.java. This class extends the action class. The additional fields present in it are
  - actor - who does the action
  - initial_object - old information
  - final_object - new information that is constructed using the old information
  - cd-form -
    (MBUILD
    (ACTOR VAR1)
    (INITIAL VAR2)
    (FINAL VAR3)
    (CONC_TENSE VAR4)
    (USING VAR5)
    (ADVERB VAR6)
)

- **MTRANS** - The corresponding java file is mtrans.java. This class extends the action class. The additional fields present in it are -
  - source - initial position
  - dest - final position
  - actor - who does the action
  - object - object that is shifted from one place to another

  - cd-form -
    (MTRANS
    (FROM VAR1)
    (OBJECT VAR2)
    (ACTOR VAR3)
    (CONC_TENSE VAR4)
    (USING VAR5)
    (ADVERB VAR6)
)

- **OBJECT_NAMES** - The corresponding java file is IObject_structure.java. This class extends the Object_structure class. The additional fields present in it are
  - token
  - name
  - cd_form -
    (OBJECT
    (NAME VAR1)
    (POSSBY VAR2)
    (ISA (ADJECTIVE VAR3))
)
  - part_of_speech (= abstract for abstract objects like decision or speech)
  - adjective - adjective of the object
  - poss_by - by whom is this object possessed by
  - size - size of the object. It is parameterised as
    * 10 : "massive
    * 9 : "huge"
    * 8 : "normal"
    * 7 : "small"
    * 6 : "tiny"
  - physical state - condition of the object. It is parameterised as
    * 10 : "ok"
    * -3 : "hurt"
    * -5 : "broken"
    * -9 : "harmed"
    * -10: "dead"
  - quality - quality of the object. It is parameterised as
    * 10: "superb"
    * 9 : "good"
    * 8 : "average"
    * 7 : "bad"
    * 6 : "poor"

- **PLACES_NAMES** - The corresponding java file is Actor_Place_structure.java. This class extends the IObject_structure class.
  - cd-form is -
    (PLACE
    (NAME VAR1)
    (POSSBY VAR2)
    (ISA (ADJECTIVE VAR3))
)

- **POLITICIANS_NAMES** - The corresponding java file is Actor_Place_structure.java. This class extends the IObject_structure class. The additional fields present in it are
  - cd-form -
    (PERSON
    (NAME VAR1)
    (POSSBY VAR2)
    (ISA (ADJECTIVE VAR3)))
  - mental_state - mental state of the person. It's parameters are -
    * 10: "ecstatic"
    * 5 : "happy"
    * 2 : "pleased"
    * 0 : "ok"
    * -2 : "sad"
    * -3 : "upset"
    * -5 : "depressed"
    * -9 :"catatonic"
  - physical_state - physical state of the person. It's parameters are
    * 10: "ok"
    * -3 : "hurt"
    * -5 : "injured"
    * -9 : "harmed"
    * -10 : "dead"
  - health - health of the person
    * 10 : "perfect"
    * 7 : "tip top"
    * 0 : "all right"
    * -7 : "sick"
    * -9 : "gravely ill"
    * -10 : "dead"
  - anger - level of anger of a person

  - 10 : "perfect"
  - 7 : "tip top"
  - 0 : "all right"
  - -7 : "sick"
  - -9 : "gravely ill"
  - -10 : "dead"

- **PRONOUN_NAMES** - The corresponding java file is Actor_Place_structure.java. This class extends the IObject_structure class. A pronoun encountered while pronoun is simply replaced with the corresponding noun learnt. Hence there is no CD form for the pronouns. PROPEL - The corresponding java file is propel.java. This class extends the action class. The additional fields present in it are
  - source - initial location
  - dest - final location
  - object - object that is moved
  - cd - form -
    ```
    (PROPEL
    (FROM VAR1)
    (TO VAR2)
    (OBJECT VAR3)
    (PHYSCONT VAR4 VAR5)
    (CONC_TENSE VAR6)
    (USING VAR7)
    (ADVERB VAR7)
    )
    ```

- **PTRANS** - The corresponding java file is ptrans.java. This class extends the action class. The additional fields present in it are actor - who does the action
  - source - initial location
  - destination - final location
  - object - object that is to be transferred
  - cd - form -
    ```
    (PTRANS
    (FROM VAR1)
    (TO VAR2)
    (OBJECT VAR3)
    (ACTOR VAR4)
    (CONC_TENSE VAR5)
    (USING VAR6)
    (ADVERB VAR7)
    )
    ```

- **SPEAK** -The corresponding java file is speak.java. This class extends the action class. The additional fields present in it are
  - source - who speaks
  - destination - to whom is the speaker talking to
  - cd - form -
    ```
    (SPEAK
    (FROM VAR1)
    (TO VAR2)
    (OBJECT VAR3)
    (CONC_TENSE VAR4)
    (USING VAR5)
    (ADVERB VAR6)
    )
    ```
  - object - what is being spoken

- **STATE** - The corresponding java file is state.java. This class extends the action class. The additional fields present in it are
  - actor - who is doing the action
  - object - on whom is the action done on
  - cd-form- for a state change -
    ```
    (CHANGE
    (OBJECT VAR1)
    (STATE VAR2
    (INITIAL VAR3)
    (FINAL VAR4)
    )
    ```

IV. Different senses of the words, and the semantic constraints on the different role fillers

Certain type of words fall under more than one category of verbs if they have multiples senses in the English language. Eg the word "hurt" is there in both PROPEL and MTRANS category of verbs. "Hurt" is used in PROPEL when the actor hits someone or something by propelling an object (Hurting with stones changing "physical state"). "Hurt" is used in MTRANS which means that the actor can hurt someone by communicating something (Eg hurting with words changing "mental state").
The different senses will have different results. These different semantic meanings when well captured, enable the parser group to distinguish between meanings of the words depending on the context.
Semantic constraints on different role fillers has been incorporated to ensure that the sentences make sense in the English Language. Eg. The statement "Ram spoke to an apple" is not meaningful

in English. The data structures have been created in such a way that the type of fields present in the structure implicitly ensure that these constraints are imposed. For example, the subject for all the actions is of Actor type as only he can perform any action. Similarly, the object in EXPEL can only be an object (sweat, blood) and not a person. Othe similar constraints have been followed in creating the dictionary.

V. Miscellaneous Points

- Indexing of the lexicons on their names is done in a hashmap which is then used by the parser. (1b)
- Words leading to state changes of the actors are accommodated appropriately. Eg kill is doing something to the object of the action that leads to the object's health state becoming -10 (dead).
- Tenses of the verbs have been added appropriately and verbs in all the tenses mentioned previously have been added.
- Words which have a primitive CD action along with an instrumental object have been added to the lexicon list. Eg walk is PTRANS using feet.

VI. How to use the code

The entire project contains -

- A list of text files containing the tokens used in the project domain. These words are parsed by the JAVA code into appropriate data structures.
- java files containing the class description for each of the separate categories of words as shown in the figure.
- java files to feed the words in the lexicon in their respective data structures. These files are
  - get_domain.java which is used to initialise the objects, nouns, adjectives and adverbs
  - get_verbs.java which is used to initialise the verbs

Running the project - krr.java file is the main file of the project and it instantiates get_verbs class type object and get_domain class type object. A make file has been included in the project source. Make has two options

- make - compiles the project and gives us the class files
- make clean - removes the class files

To run the project after making -

```
$ java krr.java
```

References

[Eugene Charniak, Drew McDermott] Introduction to Artificial Intelligence