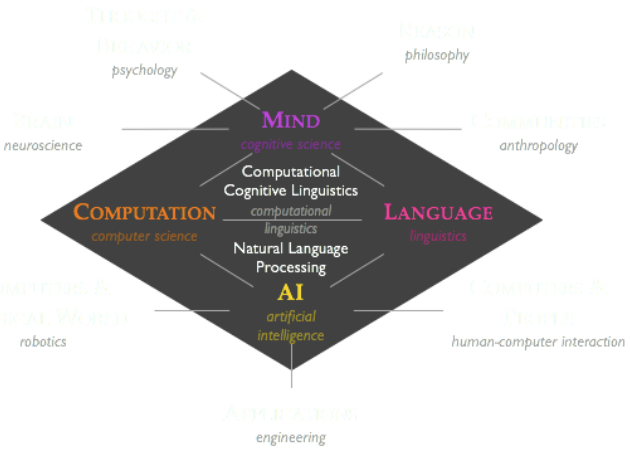
SEMESTER LONG INTERNSHIP REPORT

BUILDING A SURFACE REALIZER FOR GENERATING NARRATIVE SUMMARIZATIONS FROM ANALYZED, DYNAMIC SPATIAL-TEMPORAL DATA



ADVISOR: **PROF.DR. MEHUL BHATT**
Principle Investigator, Design-Space Group;
Professor, Informatics Department, University of Bremen;
Stiftungs Professor - German Research Centre for Artificial Intelligence (DFKI Bremen), Germany

SUBMITTED BY: **HARSHITA JHAVAR**
Semester Long Intern (Feb 1, 2015-June 30, 2015)
B.Tech. in Computer Science and Engineering
Maulana Azad National Institute of Technology, Bhopal, India

Universität Bremen cognitivesystem Santander SFB/TR 8 SPATIAL COGNITION

Acknowledgement

My sincere thanks to Prof. Dr. Mehul Bhatt for giving me the opportunity to work at Design-Space Lab, University of Bremen, Germany under his esteemed tutelage and expertise guidance. It was not without his patience and support that I was able to learn and explore an entire different sub-field of NLP ie. Natural Language Generation. It was his constant encouragement at his seminar session on Cognitive Computing which got me more interested into the cognitive aspect of Language acquisition and Translations. It is due to these seminar sessions that I have been able to conclude that for building an American Sign Language Generator, one has to understand the cognitive aspects of the Language Acquisition by the Hearing Impaired because perception is involved in the same; this is indeed my dream project to build a Natural Language-ASL Sign Language Translator and a Generator. Thus, this internship has opened doors for some new concepts and ideas in my mind and motivated me further for research through graduate studies. My special thanks to PhD student Jakob Suchan for helping me with all technical difficulties and patiently guiding me towards doing my work to a better level. It was the optimistic and cheering team at Design-Space Group consisting of my colleagues and friends at Bremen that made this internship experience abroad a complete bliss. I hope to continue my endeavours and enter my higher graduate studies as soon as possible to study my field of interest more systematically and thus, come up with technology that may help facilitating promising solutions to different concerns regarding language technology. Thanks to Santander Bank and University of Bremen for assisting me with BISIP Santander Scholarship to make my stay and work more comfortable in Germany.

Abstract

I concentrated on building a Context Free English Grammar based Surface Realizer for Generating Narrative Summarizations from Analyzed Dynamic Spatial-Temporal Data. I manually built the input data in form of prolog predicates based on the experiments conducted by the Team Project Augment; assuming a similar quality input will be generated in future; by the team which will be using the system. The task of the Surface Realizer is to select, inflect and order words to communicate the input meaning in natural language as clearly and fluently as possible in context. From each input semantic in form of prolog predicates; I focused on the production of at least one high quality output sentence. I also considered measuring the variations in generated summarizations from the surface realizer and calculated the efficiency of the system and the lexical structure quality of the generated sentences.

Contents

Chapter 1

Introduction

Natural Language Generation is a sub-field of natural language processing which in turn can be seen as a sub-field of both computer science and cognitive science. It is the process of mapping internal computer representations of information into human language. It is an area of computer science which is becoming increasingly important as the quality of computer software is judged more and more by its usability. Questions like how linguistics and domain knowledge should be represented and how communication can take place between machine and human are answered by this field. From a practical perspective, Natural Language Generation technology is capable of partially automating routine document creation, removing much of the drudgery associated with such tasks. There are various Natural Language Generators for different domains, like weather forecasting, baby health report generation etc. But there is no Natural Language Generation engine which can read analyzed dynamic spatial-temporal domain data. The scientists dealing with various visual computing, space design or image processing experiments have to either hard code the text which they intend to generate or they have to use the pre-defined templates. There exists no generic model through which data summarizations in form of narratives or textual summaries can be generated.

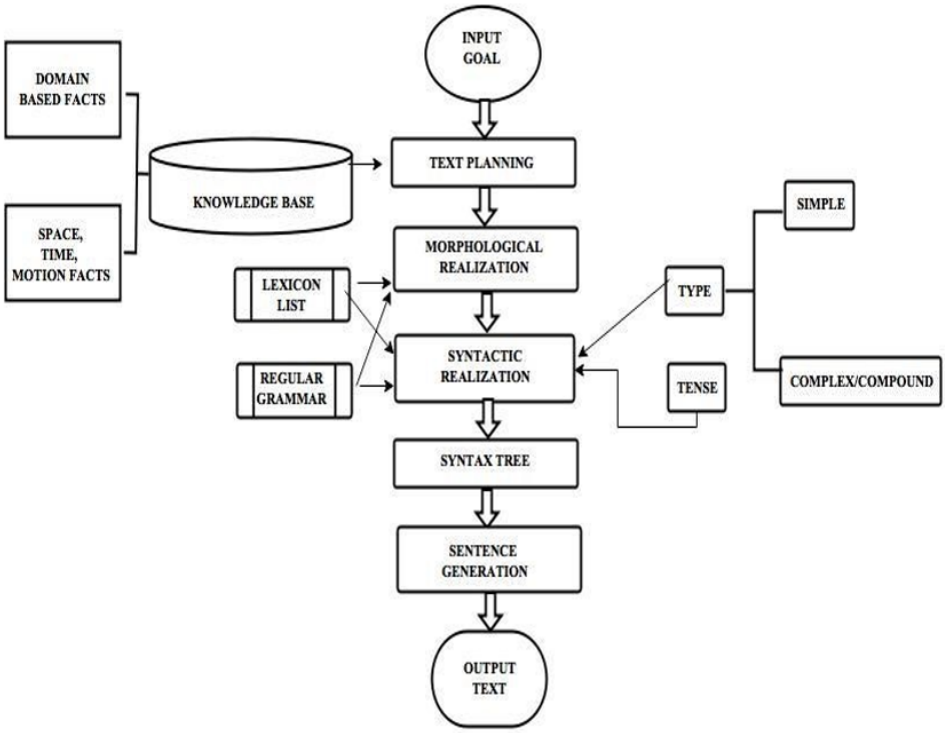
The surface realisation task consists in using a grammar to produce the sentence(s) associated by this grammar with some input meaning representation. The meaning representation is written using some logical language. For example, a surface realiser might receive an input semantics of loves(l,john,mary), for which it outputs the surface realisation John loves Mary. Surface realisation, also referred to as tactical generation, is similar to the better known task of natural language parsing. When doing generation, I construct syntactic trees which are associated with the input semantics by the languageâĂŹs grammar, and trivially read the tree leaves to reconstruct the output string. Polynomial-time algorithms for parsing have long been developed, so it would be tempting to assert that genera-

tion is parsing in reverse and simply reuse these algorithms. But we can reduce it to a surface realisation task. Both parsing and generation have the potential to be computationally expensive because natural language has a great deal of lexical ambiguity; however, parsers have the luxury of string positions. Using string positions, one can pack an arbitrary number of alternative structures into a small and fixed number of edges, so that parsing can be done in cubic time no matter what the ambiguity. The situation for surface realisers is more delicate because the input semantics have no natural order, and so there are as many potential edges as there are subsets of literals. This number is exponential with respect to input length, and as a result, we are forced to deal with the full effect of lexical ambiguity.

Chapter 2

Architecture of the Surface Realizer

Building a surface realiser requires three ingredients: a semantic representation, a formal grammar for natural language syntax augmented with a semantic dimension, and an algorithm for building syntactic structures.



Our Context Free Grammar (CFG) based Surface Realizer is a âĂŹSingle-InteractionâĂŹ mode realizer where each invocation of the system produces a single sentence in order to produce an optimum quality sentence in context. The entire system has been built declaratively in prolog. The specification of the differ-

ent types of processing in this system are distributed across a number of distinct, well-defined and easily-integrated component modules. These modules interact with each other during the process of text generation. Less complex modularization made it easier to reuse the components efficiently amongst the different applications. The production of the output utterances from this system follow the following architectural specifications of the modules, executed in response to the user inputs.

Algorithm used for building Surface Realizer: Head-driven generation

Head-driven generators are based on the observation that every syntactic tree necessarily has a semantic head, the lowest possible node whose semantics is that of the tree. In the sentence *John smells funny*, we can identify the verb *smells* as the semantic head for the verb phrase smells funny and consequently for the sentence John smells funny. The semantic head can be used to guide the generation process.Here, the semantic heads are the inputs which will be discussed in the next section. The grammar is preprocessed so that the chain rules whose semantic head is a *Non terminal value* i.e. the input data, and non-chain rules whose semantic head is its *Terminal Value*. In the example, only t1, t2 and t3 are chain rules as described in the CFG grammar.

```
s(S) --> np(NP), vp(NP,S). % t1
vp(NP,S) --> vp(NP,VP), adv(VP,S). % t2
vp(NP,S) --> v(NP,S). % t3
v(NP,smell(NP)) --> smells. % t4
adv(VP,funny(VP)) --> funny. % t5
adv(VP,nice(VP)) --> nice. % t6
np(john) --> john. % t7
```

The generator works with what is called a target node. The first target node has category s (for sentence) and the semantics equal to the input semantics. The basic approach is to use top-down processing of the non-chain rules to identify the semantic head of the target and use bottom-up processing of the chain rules to connect the two nodes. Any new substitution sites encountered along the way are recursively processed as targets. We can think of the generator as cycling between three behaviours: leaping (top-down), connecting (bottom-up) and recursion (top-down). When tracing the algorithm, it is useful to imagine a stack of target nodes. The algorithm does not actually use such a stack because it is implicit in the recursive behaviour. The head-driven approach combines the most attractive properties of bottom-up and topdown processing, namely termination and efficiency. Like bottom-up algorithms, the head driven algorithms lexically select and then consume a finite set of rules on the basis of their semantics. The main difference is that the lexical selection (leaping) is interspersed with the generation loop as opposed to being set in the very beginning, but the essential property of using the semantic information is retained. The efficiency of these approaches comes from the top-down ability to guide the selection items by syntactic criteria. Thus, the Surface Analyzer is based on the Head Driven Generation whose components are explained in detail in the coming sections.

2.1 Input Goal

The input to the realizer is a file that contains the information selected for content determination in form of prolog predicates. The input data to the realizer should ideally come from various experimental results by the visual-computing

scientists. The task of 'Content-Determination' for building a Natural Language Generator is expected to be generated in form of results from the Visual-Computing Team. For the purpose of dealing with building the second task 'Surface Realizer'; I took the videos from the Dallas Hospital Project and manually framed the sample input data files for this project. The input data structure assumed by my system is discussed in the following sections.

2.1.1 Dictionary Data File

The dictionary data file consists of all the object, subject and motion-name list which are input to the system as nouns and verbs.

```
% Noun(name,type), 25 Subjects
 noun(barbara,living).
 noun(mehul, living).

% ObjectListOfNoun(name,type), 14Objects, 10Directions
 noun(hospital, definiteNonLiving).
 noun(door, indefiniteNonLiving).
 noun('exit-sign', definiteNonLiving).

% verb(RootForm, Present, Past, Continuous)
 verb(walk,walks,walked,walking).
 verb(go,goes,went,going).
```

2.1.2 Domain Description File

The domain description describes all the spatial relations between different objects which are read by machine as preposition. This file also holds the various analyzed properties of all the object which are read by machine as adjective. It holds the various analysis values for different motions recognized by the system and are read by the machine as adverbs. These identification of different parts of speech takes place in Morphological Realization.

```
% Defining the identified Spatial Relation between the objects.

objectSpatialRelation(Subject, SpatialRelation, Object).
% Examples:
objectSpatialRelation(Subject, left, Person):- subject(Subject), person(Person).
objectSpatialRelation(Subject, towards, 'drinking-water-sign'):- subject(Subject).

% Defining the properties of different objects.

objectInformation(Subject,PropertiesOfObject, Object).
% Examples:
objectInformation('50mlong', corridor).
objectInformation(blue, elevators).
```

2.1.3 High Level Analysed Data File (Actions Identified)

This file consists of the subject information, action information and object information. The object which comes in play in a given time coordinate, the action(s) took place in a particular time coordinate are mentioned in this file. The decision of generating complex and compound or simple sentences takes place from High Level Data File. The actions in same time coordinate for different users or may be same user, are taken into complex sentence generation.

```
%Subject Name
subject(barbara).

%Properties of action identified
%actionInformation(subject, property, action, time-coordinate).

actionInformation(barbara,quickly,walk,111).

%Simultaneous Actions have same time coordinate
compound(action(barbara, pass, mehul, 112), action(mehul, reach, sign, 112)).

action(barbara,walk,corridor,11),
action(barbara, pass, mehul,11).

action(barbara, focus, sign,11),
action(barbara, turn, left,11).
```

2.2 Text Planning: Structure of the Input Text

For every single interaction obtained, the spatio-temporal domain experts alot words corresponding to the analysed data which are taken as input into the NLG engine in the following format shown below. The system tries to structure the high level data in the formats described in the upcoming section.

2.2.1 Input Structure

- Properties of the subject in the interaction
- Name of the subject in the interaction
- Analysis of the motion identified
- Name of the motion identified
- Object List

  Relation between the two consecutive Results
- Properties of the Object
- Name of The Object
- Time Unit

The properties of Subject in the interaction and the name of the Subject are concerned with the user of the experiment. The analysis and the name of the motion identified deal with the action and its properties. Object list consist of spatial relation between two objects, name and properties of that object.

```
%The system reads the High Level Analysed input in the following format after generateSummary is called.

generateSimpleSentence (  [ [ ],'Barbara', [ ], focus, on, 'Drinking Water', sign, at, corridor, end], Sentence).

generateComplexSentence( [ [ [ ], barbara, [ ], walk, in, [ ], corridor, [ ], [ ], [ ],present_continuous ],[ three,
persons, [ ], pass, to, [ ], left, [ ], [ ], [ ], simple_present],[ one, person, [ ], pass, from, [ ], right, [ ], [ ], [ ],
simple_present  ]],Sentence).
```

2.3 Linguistic Realization

This stage allows detailed grammatical knowledge to be encapsulated within the morphological and syntactic realizer as described in the following sections:

2.3.1 Morphological Realization

Morphological realization deals with the content aspect of the syntactic realization. This stage involves the following:

Lexicon The user is allowed to generate this list independently according to the domain of the analyzed data. This list consists of a list of all nouns and verbs, expressed in the form of prolog facts.

- Prototype of structure of a noun in the list.

```
noun(Property of Noun, Type of Noun).
```

o Living nouns are the names of anything that represents a living object and is involved in the experiment. Examples of living nouns in prolog used in the NLG engine are:

```
noun(barbara, living).
noun(elisa, living).
```

o Definite non-living nouns are anything that represents a specific landmark in the experiment. Examples of definite non-living nouns in prolog used in the NLG engine are:

```
noun(sign, definiteNonLiving).
noun(trashbins, definiteNonLiving).
```

o Indefinite non-living nouns are anything that represents the nouns that are talked in general but are not specific. Examples of indefinite non-living nouns used in the NLG engine are:

```
noun(door, indefiniteNonLiving).
noun(board, indefiniteNonLiving).
```

o The spatial word list, consisting of directions and locations, are also mentioned under definite non-living nouns as they are specific in their meaning at an instance and are structured to be a part of the object phrase in the NLG system. Examples of spatial word list expressed as nouns:

```
noun(right,definiteNonLiving).
noun(left, definiteNonLiving).
noun(end, definiteNonLiving).
```

- The verbs are listed as facts in the list. They are structured as:

```
verb(Root Form of Verb,Present Form of Verb,Past Form of Verb,Continuous Form of Verb).
```

The corresponding form of the verb is being identified according to the input of the type of tense. The different examples are:

```
verb(walk,walks,walked,walking).
verb(pass,passes,passed,passing).
verb(do,does,did,doing).
```

These pre-existing well-structured verbs and nouns in the system are called during the stage of Syntactic Realization when these words play according to the domain specific Grammar to come up with a sentence.

Assignment of PoS Tags to each input

The input text is structured as explained in stage of Text Planning. The various part of speech from the input text are identified in the following correspondence:

- Properties of the subject in the interaction - Adjective
- Name of the subject in the interaction - Noun

- Analysis of the motion identified - Adverb
- Name of the motion identified - Main Verb
- Object List

  Relation between the two consecutive Results - Preposition.

  Properties of the Object - Adjective.

  Name of The Object- Noun.

- Time Unit and Tense are being assigned to the variable accordingly.

Thus, all the parts of speech are identified. The constrained data structure used for input data made it easier and less complex for the NLG system to identify the various parts of speech. This information is now passed to the Syntactic Analyzer. Here, the lexicons and their types are identified during the morphological analyser. They meet the rules written for generation in form of Grammar. An example of this is given below in a prolog format.

```
generateSentence(
        [ [ ],   %%Read as Adjective
        barbara, %%Read as Noun
        quickly, %%Read as Adverb
        pass,    %%Read as Verb
        behind,     %%Read as Preposition
        [ ],     %%Read as Adjective
        Tom,     %%Read as Noun
        at,      %%Read as Preposition
        meeting, %%Read as Adjective
        room,    %%Read as Noun
      present_continuous     %% Read as Tense
      ],Sentence).
```

2.3.2 Syntactic Realization

The parts of speech identified by the Morph Analyzer from the Morphological Realization are consulted with the simple context free grammar rules. The predicates corresponding to these rules are formed in prolog. The results are appended to form a sentence. The noun and its corresponding adjective are appended which proceeds ahead to appending the determiner to frame the noun phrase. The verb and its corresponding adverb are appended which proceeds ahead to appending the auxillary verb to it to frame the verb phrase. At last, the object phrase is appended to it to frame the entire predicate phrase. Thus, these rules are used to finally append the subject and predicate to form a sentence. It is executed according to the type of sentence as identified by the input structure:

```
%For Simple Sentences:
<Sentence> ::==
      <NounPhrase> <VerbPhrase>
<NounPhrase> :==
      [<det>] <adjective*> <Noun>
<VerbPhrase> :==
      [<auxillaryVerb>] <adverb*>
      <mainVerb> [<ObjectPhrase>]
<ObjectPhrase> :==
      [<preposition>] <nounPhrase>
```

For Complex Sentences: Usage of because,after,therefore,so.

```
%Complex statement-
<Sentence> ::==
   <S> ('because')|('therefore')|
      ('so')|('after') <S>
<S> ::==
   <NounPhrase> <VerbPhrase>
<NounPhrase> :==
   [<det>] <adjective*> <Noun>
<VerbPhrase> :==
   [<auxillaryVerb>] <adverb*>
   <mainVerb> [<ObjectPhrase>]
<ObjectPhrase> :==
      [<preposition>] <nounPhrase>
```

For Compound Sentences: Usage of if-then,while-and.

```
%Compound statement-
<CompleteSentence> :==
   <``While''> <S> <<',>
   <S>+> [<'and'> <S>]
<S> :==
   <NounPhrase> <VerbPhrase>
<NounPhrase> :==
   [<det>] <adjective*> <Noun>
<VerbPhrase> :==
      <auxillaryVerb> <adverb*>
      <mainVerb> <OP>
<ObjectPhrase> :==
     <preposition> <nounPhrase>
```

The above grammar predicates are implemented in prolog. The phrases are appended to generate the complete sentence. The following predicates come into execution:

Framing Noun Phrase The determiner before the noun is obtained in consultation with the Lexicon list.

- o Living Nouns - no determiner is identified.
- o Definite Non Living Nouns - 'the' is identified as determiner.
- o Indefinite Non Living Nouns - 'a' is identified as determiner. The chosen determiner is appended with the noun and their corresponding adjective as identified by Morphological Realizer. Thus, the subject phrase is framed at this stage. Similarly, all the object phrase(s) follow the frame Noun Phrase rule to form the corresponding phrases for each object. Here each object is phrased along with its analyzed properties.

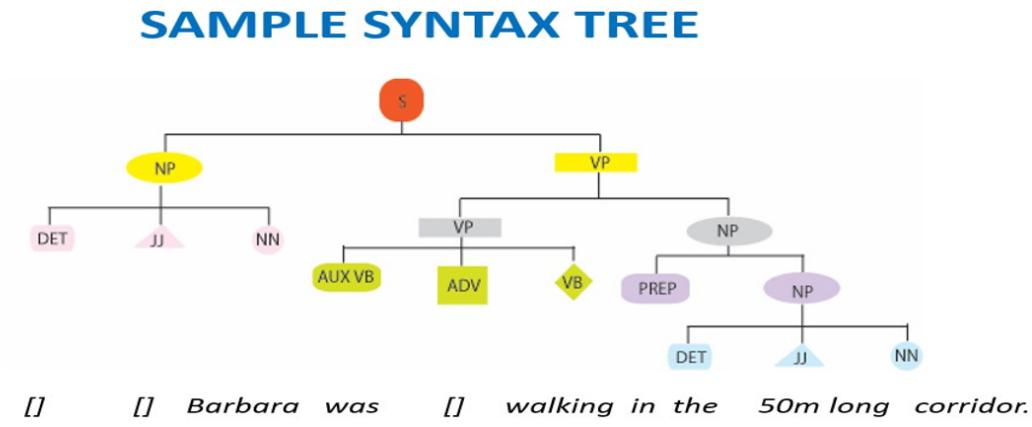
Framing Verb Phrase The verb phrases are obtained by following the input tense format and the corresponding word form. The auxiliary words are appended to the main verb. So, the verb phrase comes up with the helping verb, adverb and the verb. For complex/compound sentences, the connecting words are appended according to the grammar. They append and become a part of verb/noun phrase. Thus, the subject phrase, verb phrase and object phrase are appended to generate a syntax tree out of it. The prolog example is shown below:

```
generateSimpleSentence([little, barbara, quickly, focus, on, drinking-water, sign, at, corridor,
end, present_continuous, time_coordinate],Sentence).
% barbara is a living noun, so no determiner, sign is definite non living noun, so <the>
% is determiner, end is a definite living % noun, so <the> is determiner, focus has <is focussing>
%as its form of usage as tense is present_continuous,
%<little> + <barbara> = noun phrase
%<is> + <quickly> + <focussing> = verb phrase
%<on> + <the> + <drinking-water> + <sign> = Noun phrase
%<at> + <the> + <corridor> + <end> = Noun phrase
```

2.4 Building the Syntax Tree

The syntax tree is obtained as a result of the morphological and syntactic realization. The structure of the syntax tree looks like:

The syntax tree can be generated from the system and can be used to understand the overall structure of the lexicon arrangement used in sentence generation. The âĂŸbuiltSyntaxTreeâĂŹ predicate is responsible for the built of the tree.



2.5 The Sentence Generation

The syntax tree thus results in the expected sentence/text generation as output after the linearization of the syntax tree. The output goal of generation of text is achieved here. Some examples are shown below:

```
%Some functions invoked on calling generateSummary.
?- generateSimultaneous(
      [
      [[],barbara,[],walk,in,[],corridor,[],[],[],present_continuous],
      [three,persons,[],pass,to,[],left,[],[],[],simple_present],
      [one, person,[],pass,from,[],right,[],[],[],simple_present]
      ],Sentence
     ).
Sentence = 'While barbara is walking in the corridor, three persons passes to the left and one
person passes from the right'

?- generateDescription(junction1,
         [
         [sign,board,[],[],on,[],right,[],[],[],simple_present],
         [[],door,[],[],with,'Caution',sign,to,[],end,
           simple_present],
         [[], corridor,[],[],to,[],right,towards,'Attrium',lobby,
           simple_present]
         ],Sentence
        ).
Sentence = 'At the junction1 , there is a sign board on the right , a door with the Caution sign
to the end and the corridor to the right towards the Attrium lobby'

?- generateSentence([[],barbara,[],focus,on,exit,sign,at,
                     corridor,end,simple_past],Sentence).
Sentence = 'barbara focussed on the exit sign at the corridor end'

?- generateDependentText([[[],barbara,[],walk,in,large,corridor,[],[],[],
past_continuous],[[],barbara,[],go,to,gift,shop,[],[],[],past_continuous]],Sentence).
Sentence = 'barbara was walking in the corridor because barbara was going to the gift shop'
```

An example of summary generated from the system can be:

User has the name Barbara.
User did the recording at the Parkland Dallas hospital.
Barbara walks through the 50 metres long corridor.
While Barbara is walking through the corridor, Barbara passes through the chairs to the right of Barbara, the NO PHOTOGRAPH, COVER YOUR COUGH sign to the left of Barbara and the TOILET sign to the left of Barbara.

Barbara focusses on the Drinking Water sign at the corridor end.
While Barbara is passing Mehul on the right, Barbara sees Mehul for 2 seconds.
While Barbara is walking in the corridor, three persons pass to the left of Barbara and one person passes from the right of Barbara.

Barbara turns then towards the right of Barbara.
Barbara reaches the decision point.
The decision point has a sign board on the right of Barbara, a door with the caution sign to the end of the corridor and a corridor to the right of Barbara towards the 'Attrium Lobby'.

The sign board on right of Barbara has Women Emergency Pod, Atrium Lobby and Gold, Blue and Green Elevators.

Barbara looks around at decision point 2.
Barbara looks confused at decision point 3 and decision point 7.
At decision point 7, Barbara does not take the correct turn.
Barbara walks towards the door with 'Caution Sign'.
At decision point 7, Barbara turns right of Barbara instead of turning left of Barbara.
Barbara returns to decision point 7.
Barbara observes the sign board.
Barbara takes turn towards the door to the 'Attrium Lobby' on the right of Barbara.
While Barbara is walking towards the door, Barbara fixates on the 'Exit sign' above the door.
After Barbara pushes the door, Barbara focusses on another sign board, a long corridor, and an exit sign board.

Barbara reaches a decision point.
There is one sign board to left of Barbara, 3 doors to left of Barbara and a long corridor infront of Barbara.

Barbara walks straight through the corridor.

While Barbara is walking through the corridor, Barbara passes the 'Gift Shop' sign to left of Barbara, reception desk to the right of Barbara and takes left turn at end of the corridor.

Barbara sees a long and wide corridor with a big sign board at the end.
Barbara walks straight through the corridor.

While Barbara is walking through the corridor, Barbara passes one door to the left of Barbara, one sign board to the left of Barbara, the 'Toilet' sign to the left of Barbara and sofa chairs to the right of Barbara.

Barbara fixates on the big sign board at the end of the corridor.
The big sign board reads about the 'Blue Elevator', the 'Green Elevator', the 'Patient Relations', the 'Multi Faith Chapel', the 'Park Food Court' and the 'Park Market'.

Barbara reaches the blue elevators.
Barbara reaches the first floor through the blue elevator.
Barbara passes a reception desk.

Barbara takes left turn towards a door, Barbara sees the sign board to the left of a door with the 'Anderson-Bridge' sign above a door.

Barbara passes through door 2.
Barbara sees a long corridor.

While Barbara reaches the mid of the long corridor, Barbara sees a door to the left of Barbara with the 'Exit sign' and 'Bridge to Anderson Clinic' above the door.

Barber enters second half of the corridor.
Barbara sees the exit sign at the end of the corridor.
Barbara reaches the Pharmacy at the end of the corridor.
Barbara took 15 minutes to reach the destination of Barbara.

Chapter 3

Efficiency and Emperical Evaluation Tests Results

3.1 Efficiency Test

For the empirical test, we built a spatio-temporal domain based, moving image analysed data corpus. The corpus consists of 25 subjects, 500 interactions, 53 spatio-temporal domain based facts. We analysed generation time of 25 summaries on two basis - different tenses and type of clause structure of generated sentences for each summary(simple, complex, compound). These figures suggest that the spatio-temporal domain based, declarative model built for language generation, is highly robust to render paragraphs in less than a second.The time for computation was noted for different tenses and different clause structures. The average time of generation of these records is then further computed.

3.1.1 Results

- With an average of 20 interactions, on an average 26.2 sentences per summary, with 17.6 tokens as the average length of each sentence, the system took average time of 77.8 msec and 84.48msec for generating summaries in simple present/past/future and continuous present/past/ future respectively for all subjects.The subject name and its number of interactions were noted. The number of simple, complex and compound sentences in the generated summary were noted. The total length and the average length of the summary was computed. The total time and the average time of the summary was computed.

- Total length of the summary = No. of simple sentences+ No. of compound sentences + No. of complex sentences.

- The average time taken for generation of sentences with different clause structures on a scale of 100 generated sentences was found. The time for one simple sentence being approximately 0.52/msec, one compound sentence being 1.23/msec and one complex sentence being 1.32msec.

- The test was done on ubuntu 14.04, IntelÂő CoreTM i7-3630QM CPU @ 2.40GHz X 8.

3.2 Emperical Evaluation Tests

For the course of Empirical Evaluation, we found that there exists no Evaluation methods on which a Surface Analyzer can be evaluated. The implemented model is such that primitives pertaining to space and motion are available as first-class objects with deep semantics suited for inference and query. There are some well known tools and measures like the BLEU Score Technique, The ROGUE which can be used to evaluate the contextual coverage of the generated Summarizations. During the design of the Surface Analyzer, we manually built a corpus for testing purposes. So, the contextual coverage is 100% as there is a quality sentence generated for each input well-covering the context of the input. The Declarative Logical Framework nature of the system gave it cent percent contextual coverage. It might be this reason only that why

NO. OF COMPOUND SENTENCES IN SUMMARY	NO. OF COMPLEX SENTENCES IN SUMMARY	TIME FOR SUMMARY IN SIMPLE PRESENT, PAST OR FUTURE	TIME FOR SUMMARY IN CONTINUOUS PRESENT, PAST OR FUTURE	TOTAL LENGTH OF SUMMARY (Average Length of Sentence is 17.6 tokens)
3	8	79	76	24
3	7	73	92	20
3	16	88	91	30
4	6	72	99	25
3	8	73	89	27
2	4	80	81	17
3	21	71	84	39
3	6	76	82	19
3	8	82	73	24
3	17	73	78	33
3	12	81	92	30
3	8	75	86	24
2	11	76	78	24
3	8	79	76	26
3	11	70	89	28
3	8	76	88	24
3	8	87	87	24
3	11	72	81	27
3	14	85	95	27
3	15	96	89	31
3	15	81	83	30
4	7	71	76	24
3	8	76	81	21
3	11	71	82	28
3	16	82	84	29
		Average=77.8	Average=84.48	Average Length of Summary= 26.2
3	10.56	Min=70	Min=73	Min=17
		Max=96	Max=99	Max=39

SimpleNLG: another realiser Engine built under Dr. Ehud Reiter mentions only about an Efficiency Test in its paper and not any Emperical Evaluation.

Emperical evaluation can be done only when a context is being determined by the system but for a realiser engine; this is not the case because a realiser engine deals with the grammatical structure of the sentences. IBMâĂŹs BLEU metric or ROGUE metric, designed for evaluating machine translation contextual quality, scores candidate sentences by counting the number of n-gram matches between candidate and reference sentences. It also punishes differences in length between candidate and reference sentences.So, as far as the contextual coverage or the BLEU Score or the ROGUE score of the system is concerned, it is 100% and 1 and 1 respectively.

In order to still evaluate the quality of the generated summarizations based on the sentences which do not need any change in future; we can try estimating the distance between a human written report and a system generated report. Clearly, the system currently is a surface realizer and thus, needs further the implementation of Discourse theory and Lexical Aggregation for matching human quality summaries completely and thus, building a Natural Language Generator for the system.

3.2.1 Results

Here: UserCase: The different subjects. Length of Machine Generated Summary: No. of sentences generated in each summary (Simple + Complex+ Compound sentences). Correct lexical Sentence Structures: Comparing it with manually written summary to obtain the distance from the required quality of summary. Score: The range of score is 1-10. It is 10 if all the sentences in the machine generated summarization are with correct lexical structures and demand no further improvement.

For example:

Machine Generated Lexically Improperly Structured Sentence: *While Barbara reaches the mid of the long corridor, Barbara sees a door to the left of Barbara with the 'Bridge to Anderson Clinic' sign above the door.*

Human Written Sentence: *While Barbara reaches the mid of the long corridor, she sees a door on her left with the 'Bridge to Anderson Clinic' sign above it.*

Score = (Correct lexical Sentence Structure/ Length of Machine Generated Summary)*10.

Considering the user test cases and their estimated closeness to a human generated report:

UserCase	Length of Machine Generated Summary	Correct lexical sentence structures	Score
Barbara	42	25	5.9
mehul	20	14	7.0
jakob	30	20	6.67
swati	25	16	6.4
bob	27	21	7.77
tim	17	9	5.3
elisa	39	30	7.6
george	19	6	3.15
kin	24	15	6.25
raj	33	24	7.2
ana	30	19	6.33
javi	24	15	6.25
jimmy	24	17	7.08
tom	26	16	6.15
meera	28	19	6.78
ram	24	18	7.5
rashi	24	14	5.83
jack	27	19	7.03
john	27	21	7.77
joy	31	28	9.03
jason	30	21	7.08
tina	24	17	7.08
hoYa	21	15	7.14
shiv	28	21	7.5
harshita	29	21	7.24
Average Score			6.76

Thus, the average score is 6.76 which can be further improved after implementing the Discourse Representation Theory and Lexical Aggregation. The most interesting thing for a Surface Realizer is that the input to it is the already chosen content and thus, corresponding to every chosen content there is a quality output to it. Therefore, even assigning weights to different n-gram models of computation; the contextual accuracy will always be 100% as seen through ROGUE and BLEU score.

Chapter 4

Conclusion and Future Work

Conclusion

The NLG engine developed during the course of the internship is based on dynamic spatio temporal data domain. This engine is highly scalable. It can hold any length of data. The same predicates which are defined in prolog can be reused for generating more complex functionalities. The system can render paragraphs in seconds. It has reduced a lot of human effort. The same data input is able to produce different reports for different users. By reducing or increasing the amount of details, this is achievable. The system is efficient to generate analysis summaries and narrative reports in different tenses involving different clause structures of simple, complex, compound.

Future Work

Coming up with âĂŸDiscourse ModelâĂŹ for language generation in the same domain. Sentences with Pronouns, Numbers (Singular/Plural). Forming Interrogative, Reflexive, Exclamatory, Negation. Increase the features of Grammar Involved. Building Natural Language Based, Declaratively built, Question-Answering System.

References

[1] D. Roy and E. Reiter. 2005. *Connecting language to the world, Artificial Intelligence,Volume 167, Issues 1âĂŞ2, September 2005, Pages 1âĂŞ12.*

[2] M Mitchell, K van Deemter, E Reiter (2010). *Natural Reference to Objects in a Visual Domain. Proceedings of INLG-2010, pages 95-104.*

[3] Yuan ren, Kees van Deemter, Jeff Z. Pan. *Charting the potential of description logic for the generation of referring expressions. Proceedings of INLG-2010.*

[4] Barbara Di Eugenio, Michael Glass, Michael J. Trolio, Susan Haller. *Simple Natural Language Generation and Intelligent Tutoring Systems. Proceedings of the 2005 conference on Artificial Intelligence in Education: Supporting Learning through Intelligent and Socially Informed Technology.*

[5] Albert Gatt and Ehud Reiter, 2009, *SimpleNLG: A realisation engine for practical applications, ENLG '09: Proceedings of the 12th European Workshop on Natural Language Generation.*

[6] R Kutlak, C Mellish, K Van Deemter, 2013, *Content Selection Challenge-University of Aberdeen Entry, ENLG '13 Proceedings of the 14th European Workshop on Natural Language Generation*

[7] Jette Viethen, Margaret Mitchell, Emiel Krahmer 2013, *Graphs and Spatial Relations in the Generation of Referring Expressions, ENLG '13 Proceedings of the 14th European Workshop on Natural Language Generation.*

[8] Ehud Reiter and Robert Dale, 2000, *Building Natural Language Generation Systems, Cambridge University Press, Cambridge, UK.*

[9] R Kutlak, C Mellish, K Van Deemter, 2013, *Content Selection Challenge- University of Aberdeen Entry, ENLG '13 Proceedings of the 14th European Workshop on Natural Language Generation*

[10] Sina ZarrieÃČÅÿ Kyle Richardson, 2013, *An Automatic Method for Building a Data-to-Text Generator, ENLG '13 Proceedings of the 14th European Workshop on Natural Language Generation*

[11] Jette Viethen, Margaret Mitchell, Emiel Krahmer, 2013, *Graphs and Spatial Relations in the Generation of Referring Expressions, ENLG '13 Proceedings of the 14th European Workshop on Natural Language Generation.*