An intelligent co-reference resolver for Winograd schema sentences containing resolved semantic entities

by

Tejas Ulhas Budukh

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved September 2013 by the
Graduate Supervisory Committee:

Chitta Baral, Chair
Kurt VanLehn
Hasan Davulcu

ARIZONA STATE UNIVERSITY

December 2013

ABSTRACT

There has been a lot of research in the field of artificial intelligence about thinking machines. Alan Turing proposed a test to observe a machine's intelligent behaviour with respect to natural language conversation. The Winograd schema challenge is suggested as an alternative, to the Turing test. It needs inferencing capabilities, reasoning abilities and background knowledge to get the answer right. It involves a coreference resolution task in which a machine is given a sentence containing a situation which involves two entities, one pronoun and some more information about the situation and the machine has to come up with the right resolution of a pronoun to one of the entities. The complexity of the task is increased with the fact that the Winograd sentences are not constrained by one domain or specific sentence structure and it also contains a lot of human proper names. This modification makes the task of association of entities, to one particular word in the sentence, to derive the answer, difficult.

I have developed a pronoun resolver system for the confined domain Winograd sentences. I have developed a classifier or filter which takes input sentences and decides to accept or reject them based on a particular criteria. Once the sentence is accepted. I run parsers on it to obtain the detailed analysis. Furthermore I have developed four answering modules which use world knowledge and inferencing mechanisms to try and resolve the pronoun. The four techniques I use are : ConceptNet knowledgebase, Search engine pattern counts,Narrative event chains and sentiment analysis. I have developed a particular aggregation mechanism for the answers from these modules to arrive at a final answer. I have used caching technique for the association relations I obtain for different modules so as to boost the performance.

I run my system on the standard 'nyu dataset' of Winograd sentences and questions. This dataset is then restricted by my classifier to 90 sentences. I evaluate my system on the 90 sentences' dataset. When I compare my results against the state of the art system on the same dataset, I get nearly 4.5 % improvement in the restricted domain.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor Dr. Chitta Baral for his valuable suggestions and guidance throughout my thesis. He taught me a lot of things about research. I would also like to thank all my colleagues in BioAI lab for their support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction and Motivation

There has been a lot of debate on whether machines could think like a human or not. Alan Turing proposed a test that focuses on observable behaviour of a machine to solve this problem. The goal of this test was to have a machine participate in a natural language conversation with the person. But the trouble with Turing test $(Turing, 1950)$ is that a machine might try deception or some sort of trickery and fool the person into believing that he is talking to another person. The other alternative test, that is proposed recently in 2011, is "The Winograd Schema" challenge($Levesque,\ Davis,\ Morgenstern$). It is a variant of RTE(Recognizing Textual Entailment) challenge ($Dagan,\ Glicksman,\ and\ Magnini\ 2006;\ Bobrow\ et\ al.\ 2007;\ Rus\ et\ al.\ 2007$). RTE challenge involves a machine answering yes/no type of questions concerning whether one English sentence A entails another sentence B, as illustrated with example below.

Example 1 *Example :*
A: Time Warner is the world's largest media and internet company.
B: Time Warner is the world's largest company.

Some reasoning ability and background knowledge is required to solve this challenge. But this seems to be below the difficulty level, of what Turing test expects.

Hence a variant of RTE, The Winograd Schema Challenge, is proposed.It requires the subject to answer binary questions which require reasoning ability and background knowledge or real world knowledge to draw inferences, but it doesn't explicitly depend on the entailment of the sentences. Answering these questions involve complex cases of co-reference resolution, for which commonly used methods like string-matching are not useful. It tests for all the properties, that Turing desired in an intelligent system.

The coreference resolution as mentioned above is actually a pronoun resolution task. The pronoun is also called as a difficult pronoun. There is one difficult pronoun in the sentence and two candidate entities. The pronoun refers to one of the two candidate entities or candidate answers and we have to find out which candidate answer the pronoun resolves to. Traditionally used approaches on co-reference resolution don't work in this case because this problem has more complex sentence structure which requires more sophisticated semantic parsers, knowledge bases and inference mechanisms, to find an answer. The goal of our research is to build an intelligent coreference resolver system for The Winograd schema challenge, using various commonsense knowledgebases.

1.1 The winograd Schema Challenge

The Winograd schema challenge is a restricted application of, a more complex complex, pronoun resolution problem. A Winograd schema contains a binary question, i.e. a question which has two possible answers in the sentence, but only one of them is a correct one. An Example is :

Example 2 *Joan made sure to thank Susan for all the help she had given. Who had given the help ?*

- *Joan*
- *Susan*

The answer is obvious to the human reader here but not to the machine. The answer can't be found using the other commonly used techniques like, String matching, selectional restrictions or statistical techniques over text corpora. Each of the Winograd sentence has following properties :

1. Two entities are mentioned in a sentence by noun phrases(NP). The two entities don't have any fixed gender nor can they always be classified as instances of entities such as animals, people etc. Many a times these two entities are referred by their proper names, if they are people, which makes resolution harder.

2. A pronoun or possessive adjective is used in the sentence in reference to one of the parties, but is also of the right sort for the second party. The pronoun used, in the sentence, can be one of the $he, him, his, she, her, it, its, they, them, their$.

3. The question involves determining the referent of the pronoun or possessive adjective. This is the coreference resolution problem mentioned above in the introduction. There are two

answers possible – Answer 0 is always the first entity mentioned in the sentence, and Answer 1 is the second entity.

4. There is a special word that appears in the sentence and may be the question. When it is replaced by another word (called the alternate word), the sentence still makes perfect sense but the answer changes.

The peculiarity of a Winograd sentence is that it can be about anything, it has no limitations on the other additional noun phrases present in the sentence or other pronouns that can be included in the sentence. Although there is a restriction on the vocabulary to be used in the sentence, that it would be restricted enough that even a child can understand the question. The alternate version of the above given example is :

Example 3 *Joan made sure to thank Susan for all the help she had received. Who had received the help ?*

- *Joan*
- *Susan*

This sentence has all the above mentioned properties and the special word is : given, the alternate word is : received.

Because of all the above mentioned properties, the Winograd schema is called "Google Proof" i.e. only having large text corpus in hand is not going to help machine answer the Winograd question. Google proof also means that there is no obvious statistical method, that can be applied to the large text corpus, to derive an answer. The Winograd schema makers claim that, for doing any better than just randomly guessing the answer, you have to be able to figure out what is going on in the sentence i.e. you have to be able to draw inferences. For example : in the above sentence you have to be able to make an inference that the one who receives help would thank someone who gives help. It is not at all possible that there will be any statistical or any other specific properties of a special word or an alternate word that would change the answer when given is changed to received. Hence you need to have background knowledge to resolve the situation and infer the answer.

1.2 Motivation

There has been a lot of work in the pronoun resolution domain. That work can mainly be classified into two main approaches, Knowledge-based and statistical. The challenge that Winograd schema offers is knowledge based : "explicitly representing knowledge in a formal language and provide procedures to reason with that knowledge"(Brachman and Levesque 2004). Statistical approaches don't simply provide the type of deep reasoning required for solving Winograd schema challenge. They can however be used to gather some commonsense knowledge or common patterns of pronoun referents, when used over a large text corpus. The use of statistical approaches to learn common patterns in text combined with some reasoning/inferencing capability can provide an interesting approach to solve this problem.

The main motivation behind this research is that the Winograd schema makers proposed this challenge as an alternative to the famous Turing test $(Turing, 1950)$. And solving this challenge successfully will take us a step forward towards intelligent 'thinking' machines. There are several commonsense knowledgebases being made available by many researchers. We try to use these knowledgebases to extract world knowledge and draw inferences from them. We try to take forward this new line of research, opened up by $Rehman, ng(2012)$ in their research for resolving complex cases of definite pronouns.

The winograd schema corpus

While designing Winograd schema, the makers try to avoid two main pitfalls, that arise while making the answer obvious to the human readers($Levesque,\ 2011$). The two pitfalls are as follows :

- Levesque(2011) defines Winograd schema as a "small reading comprehension test, with one question, in which one of the two candidates in the question correctly resolve to the definite pronoun present in the comprehension". Levesque tried to design the Winograd schema according to this definition, in such a way that the background knowledge required for resolving the definite pronoun to correct candidate is not presented in the given sentence. In accordance to this definition, the schema makers try to avoid a pitfall concerning the questions whose answers are too obvious. For example :

Example 4 *The racecar zoomed by the school bus because it was going so slow/fast. What was going so slow/fast ?*

- *Racecar*
- *Bus*

In this example the association between racecar and speed is too obvious, and hence it provides a very strong hint about the answer of the question. The schema makers try to avoid this pitfall by replacing 'racecar' with a 'delivery truck'.
Also in case of a person type of entity in the sentence, they try to avoid this pitfall of making the answer too obvious, by using proper names for the candidate answers. In this case there is no chance of connecting the proper names with the question by making associations. For example :

Example 5 *The mother gave birth to a little daughter, she was a very charming woman. Who was a charming woman ?*

- *Mother*
- *Daughter*

In the above example the answer is too obvious because the phrase 'charming woman' can be easily associated with the word 'mother'. Hence the Winograd schema makers use proper names instead of semantic roles. The new sentence will be as follows :
Mary gave birth to a little Ann, she was a very charming woman. Who was a charming woman ?

- *Mary*
- *Ann*

In this case, the proper names $Mary$ and Ann have to be first resolved to their semantic roles – $Mother$ and $Daughter$ before we can make associations.

- The second pitfall is about keeping the vocabulary of a Winograd sentence simple enough, to understand the situation easily. However choosing a special word and its alternate word from the simple vocabulary, while not making the sentence ambiguous, is a difficult task. The schema makers try to avoid this pitfall by providing more information on the candidates, if the vocabulary becomes too rich. For example :

  Example 6 *Frank felt vindicated/crushed when his longtime rival Bill revealed that he was the winner of the competition. Who was the winner ?*

  - *Frank*
  - *Bill*

  The additional information, that Bill was a longtime rival of Frank, is provided to make it clear in the sentence that Frank was the winner in case of vindicated as a special word.

1.3 Main Contribution

In this research we have implemented a pronoun resolver system that takes as input, a sentence containing two candidate antecedents and a pronoun, and a question. The question targets to resolving the pronoun to one of the candidate antecedents in the sentence. We have used several external libraries for various purposes, that will be explained in the chapters to follow. We have implemented a sentence parsing phase which parses the given sentence and a question,using two parsers, and extracts desired syntactic components from them, which will be used in the further processing phase. We try to make sentence parsing and answering phases faster, by implementing several efficiency improvement techniques such as caching the earlier results using Hash-Maps, parallel processing independent components, keeping data in more dynamic memory structures as opposed to the files on the disk etc. We have used Java programming language for our implementation. For some of the memory heavy tasks, like finding paths in ConceptNet schema, we need some logical programming language that can perform this task easily and faster, hence we use Answer Set Programming (ASP) with the help of Clingo. This is explained in detail in chapter 3. We have also made use of WordNet database[1] that is locally installed on the system on which the program runs. We have implemented an inbuilt parser for the locally downloaded FrameNet database[2]. We have used another freely available knowledgebase ConceptNet[3], locally. We have created our own duplet dataset from it and fire search queries on that, but we also use ConceptNet project's online serch API[4] and association API[5]. We evaluate our pronoun resolver on the original Winograd schema sentences' corpus[6]. Since we receive answers from three different knowledgebases, we have implemented a classification and aggregation mechanism which allows us to come up with one final answer. We test the answers from our system against the given answers.

1.4 Thesis outline

We have already seen the introduction and motivation behind designing the pronoun resolver system. Rest of our work is organized as follows : Chapter 2 will provide insight into the sentence parsing for a given Winograd sentence. We have used two external parsers and implemented a system to extract desired syntactic components from them. This chapter will provide details about it. Chapter 3 provides the implementation details of how we extract world knowledge from existing knowledgebases and use it to draw inferences for resolving the pronoun to one of the candidate antecedents. It also provides the details of our result classification and aggregation mechanism. Chapter 4 provides details of our results and system evaluation on classified Winograd schema chosen from the original winograd schema by Levesque(2011). Chapter 5 finishes the work with a conclusion and remarks about future work to improve our system.

[1]https://wordnet.princeton.edu/wordnet/download/
[2]https://framenet.icsi.berkeley.edu/fndrupal/about
[3]http://conceptnet5.media.mit.edu/downloads/current/
[4]http://conceptnet5.media.mit.edu/data/5.1/search
[5]http://conceptnet5.media.mit.edu/data/5.1/assoc
[6]http://www.cs.nyu.edu/faculty/davise/papers/WS.html

Chapter 2

Sentence Parsing

Sentence parsing is important to understand the syntactic structure of the sentence and obtaining Part Of Speech(POS) tags. We use sentence parsing for, identifying events in the sentence, obtaining subjects and objects or agent and patients of those events, obtaining POS tags of all the words in the sentence. We perform parsing of both the question and the given Winograd sentence. We also have a module to identify the pronoun to be resolved and candidate answers, from the given sentence. But since that is not yet completely developed and made foolproof, we take candidate answers and pronoun to be resolved as inputs from the user. We use two completely different types of parsers for our purpose, first one is a Boxer that uses CCG (Combinatorial Categorical Grammar) and the second one is a Stanford parser that uses PCFG(Probabilistic Context Free Grammar). We use two parsers so as to predict, the events, their order in the sentence, the participation of entities in them, more accurately.

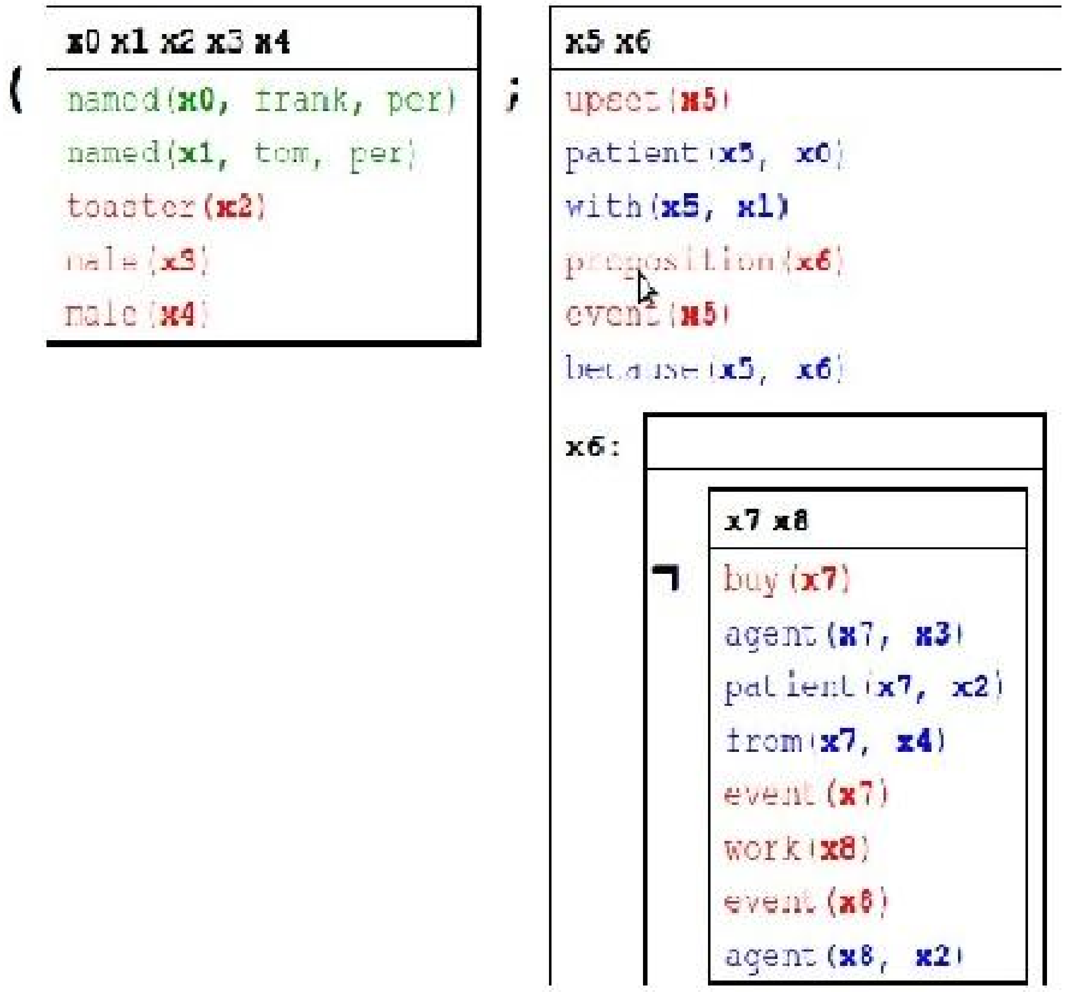
2.1 Boxer CCG Parser

Boxer is a semantic parser. It has two stages, first stage produces the intermediate CCG derivation of a sentence and then second stage works on it to produce the semantic representation known as DRS : Discourse Representation Structure(Kamp and Reyle, 1993). It detects events with the neo-Davidsonian analysis of the sentence. With this analysis it uses $proto$, by default, to detect the thematic roles of noun phrases participating in the event in the sentence. The role inventory using $proto$ consists of mainly $agent,\ patient\ or\ theme$ type of roles.

We use Boxer to produce the output of sentence parsing in the XML format, then we parse that XML using our Java program. In the output XML the Boxer produces DRSs in XML format and it also produces the POS(Part Of Speech) tags for each and every word in the input. We use these POS tags in our project to detect the use of a given word in given sentence. The DRSs are present in the form of XML tags and references, we have a reference extraction module to construct back the original DRSs from the XML. From the constructed DRSs we construct event maps and event order sequence in our program. The event map is a HashMap with event as the key and an array of Strings containing agent and patient as value. An example of event map constructed from the boxer output of a sentence is as follows :

Example 7 *Frank was upset with Tom because the toaster he had bought from him did not work .*

Boxer output with DRSs :



```
x0 x1 x2 x3 x4
named(x0, frank, per)
named(x1, tom, per)
toaster(x2)
male(x3)
male(x4)
;
x5 x6
upset(x5)
patient(x5, x0)
with(x5, x1)
proposition(x6)
event(x5)
because(x5, x6)
x6:
    ¬
      x7 x8
      buy(x7)
      agent(x7, x3)
      patient(x7, x2)
      from(x7, x4)
      event(x7)
      work(x8)
      event(x8)
      agent(x8, x2)
```

Figure 2.1: Boxer Parser Output

```
Event map from the output :

event   agent       patient

work    toaster     null

buy     male        toaster

upset   null        frank


Event order sequence :

1 upset

2 buy

3 work
```

The advantage of Boxer parser is that in the simple structured sentences it detects the events and agents/patients more accurately than Stanford parser.

The disadvantage with the Boxer parser is that the quality of its output is not certain, that means it's not a very stable parser yet. The Boxer makes mistakes while computing the output for long, complex structured sentences. Also it doesn't correctly analyse several expressions where time or measure related to some quantity is given. Hence we make use of the Stanford parser along with Boxer to correctly construct the event maps.

2.2 Stanford PCFG Parser

Stanford parser is a statistical parser. It uses knowledge of language learned from the hand-parsed sentences to produce most likely analysis of the current input sentence. It produces more consistent and reliable output for even the long or syntactically complex structured sentences. The disadvantage of Stanford parser as compared to Boxer is that it doesn't have any specific notion of an event in the sentence. But it still detects the subjects or objects of verbs. Stanford parser also produces POS tags for the input sentence. We use Stanford parser mainly to detect subjects and objects of word. We use standard Java API provided by Stanford parser. The API produces POS tags as well as typed dependencies. The typed dependencies are in the format $predicate(w1, w2)$. If the predicate contains substring 'subj' then we assume word $w1$ as an event and word $w2$ as its subject. similarly if the predicate contains substring 'obj' then we assign $w1$ as an event and word $w2$ as its object. We make the word an event, if it is not tagged as a preposition by the Stanford parser. If that word $w1$ is tagged as a preposition(POS tag : IN) then we try to assign its auxiliary subject or object to the previous available event and if they are not available because they already have their own subjects/objects, then the next available event. This is a similar process that Stanford parser itself performs while collapsing the typed dependencies. From the Stanford parser output we prepare an event map and an event order map. The event map contains an event as the key and array or strings, which contains subject or object of the event, as the value. The example of sentence parsing with the help of Stanford parser is as below :

Example 8 *Frank was upset with Tom because the toaster he had bought from him did not work .*

Stanford Parser output :

```
Typed dependencies
nsubjpass(upset-3, Frank-1)
auxpass(upset-3, was-2)
root(ROOT-0, upset-3)
prep(upset-3, with-4)
pobj(with-4, Tom-5)
mark(work-16, because-6)
det(toaster-8, the-7)
nsubj(work-16, toaster-8)
nsubj(bought-11, he-9)
aux(bought-11, had-10)
rcmod(toaster-8, bought-11)
prep(bought-11, from-12)
pobj(from-12, him-13)
aux(work-16, did-14)
neg(work-16, not-15)
advcl(upset-3, work-16)
```

```
Stanford POS tags :

Frank/NNP was/VBD upset/VBN with/IN Tom/NNP because/IN the/DT

toaster/NN he/PRP had/VBD bought/VBN from/IN him/PRP

did/VBD not/RB work/VB ./.


Event map from the output :

event   subject    object

work    toaster    null

upset   Frank      Tom

bought  he         him


Event order map:

1. work

2. upset

3. bought
```

2.3 Parsing the question

We also use the above two parsers to parse input question.

Boxer

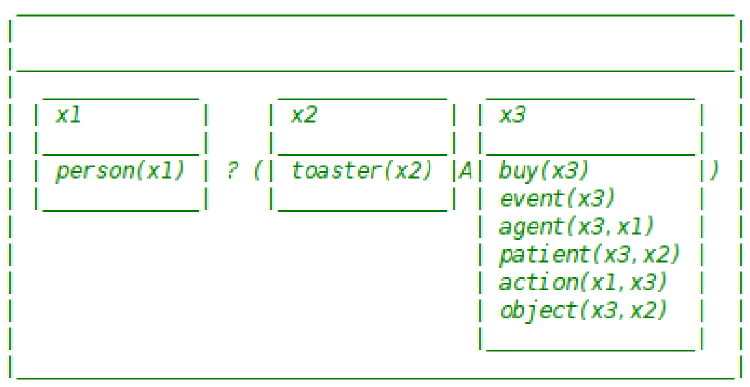
The boxer will treat questions in a different way than the normal sentence. It will try to detect an entity type from the question and map it as an agent or patient to the event in the question correctly. But since it is trained on penn TreeBank corpus which contains fewer examples of questions, it is not much reliable in terms of question parsing.

Stanford Parser

The Stanford parser analyses question in the same way as the normal sentence but the 'wh' word will be output as the subject or an object of some event.

Example 9 *Who had bought the toaster ?*

Boxer :



```
 ____________________________________________________
|                                                    |
|____________________________________________________|
|  ___________        ____________   _______________  |
| | x1        |      | x2         | | x3            | |
| |___________|      |____________| |_______________| |
| | person(x1) | ? (| toaster(x2) |A| buy(x3)       |) |
| |___________|      |____________| | event(x3)     | |
|                                   | agent(x3,x1)  | |
|                                   | patient(x3,x2)| |
|                                   | action(x1,x3) | |
|                                   | object(x3,x2) | |
|                                   |_______________| |
|____________________________________________________|
```

Figure 2.2: Boxer Parser Output

```
Event Map :

event   agent       patient

buy     person?     toaster



Stanford Parser Output:

nsubj(bought-3, Who-1)

aux(bought-3, had-2)

root(ROOT-0, bought-3)

det(toaster-5, the-4)

dobj(bought-3, toaster-5)


Event Map :

event   subject     object

bought  Who         toaster
```

2.4 Finding Candidate Answers and pronoun to be resolved

From the above discussion, it can be seen that once we parse a Winograd sentence and the question, we have event maps pertaining to them parsed by both the parsers. We try to make use of the event maps in finding the candidate answers from the sentence. However this technique is not guaranteed to work all time, hence we don't extensively use it in our project. We use this technique only for a few sentences where the question has a simple syntactic structure. To try to find candidate answers we use Boxer's XML output. In the XML output from Boxer, the person or location or organization entities in the sentence will be tagged differently within <netag> tag. So we check Boxer output for this particular tag first, if not present then we take all the entities in the sentence that are tagged as proper noun(NNP or NNS). This second method has a disadvantage that if there are more than 2 entities tagged as proper noun in the sentence, which is a case for many of the sentences, then it will detect all of them as the candidate answers. This is good for the generalized system in which many candidate answers can be present, but since our system is Winograd schema specific, we assume that there are only two candidate answers. Hence the second method of finding candidate answers doesn't work if there are some other proper nouns present in the sentence. We employ three techniques in our project to try and find the pronoun to be resolved.

Using Boxer event maps

From the Boxer parse of both the sentence and the question we have event maps for both of them. When parsing the question, the Boxer resolves the 'wh' word to an entity and appends a '?' sign to it. Hence when trying to find the pronoun to be resolved we try to find the entity with a trailing '?', and then find the event e that it corresponds to in the question event map. Then we try to find the same event e in the answer event map. If the event is present then we find the agent or patient of the event, depending on which of them has a trailing '?' in the question event map. Then from the XML output of Boxer, we find out to which word from the sentence the found agent or patient maps. Now if that agent or patient is tagged by POS tagger as PRP then we identify it as the pronoun to be resolved. Continuing the above example :

Example 10 *Frank was upset with Tom because the toaster he had bought from him did not work .*

```
Event map from the output :
```

event	agent	patient
work	toaster	null
buy	male	toaster
upset	null	frank

Who had bought the toaster ?

Event Map :

event	agent	patient
buy	person?	toaster

As stated above, the entity with trailing '?' in the question parse is : $person?$. The event in which it is involved as agent is : buy. Hence we find the same event in the sentence's parse and its agent is $male$. This agent refers to 'he' in the sentence. This information is obtained from the XML output of Boxer's parse. Hence we can detect that the pronoun to be resolved in the sentence is 'he'.

Using Stanford event maps

This process is similar to that of Boxer. The difference is that the Stanford parser doesn't resolve the 'wh' word to any entity and it also doesn't put a trailing '?' sign in front of the entity in question. So we have prepared a question type list which contains all the question types that appear in the Winograd schema. e.g. : "who", "which". So we assume that whichever event contains its subject or object as one of the entity from this list, is the event e in question. Then we go to the sentence and try to search the same event in its event map. If the event is found in the sentence event map, then we identify its subject/object as the probable pronoun to be resolved, depending upon whether 'wh' word is subject/object of the event e in question. If the Stanford POS tagger has tagged this probable pronoun to be resolved as PRP then we finalize it as the pronoun to be resolved. We illustrate this process with the above example as :

Example 11 *Frank was upset with Tom because the toaster he had bought from him did not work .*

```
Event map from the output :
event   subject    object
work    toaster    null
upset   Frank      Tom
bought  he         him
```

Who had bought the toaster ?

```
Event Map :
event   subject    object
bought  Who        toaster
```

As stated in above process, the entity with trailing 'wh' in the question parse is : who*. The event in which it is involved as subject is :* $bought$*. Hence we find the same event in the sentence's parse and its subject is* he*. Hence we can detect that the pronoun to be resolved in the sentence is 'he'.*

Using String matching

This uses simple string matching to try to find out the pronoun. It tries to match all the words from question string to the answer string except for the 'wh' word. If all the other words match then it matches 'wh' word with the word in the sentence string, at a position before the matched string. This is taken as a probable pronoun to be resolved. If the POS tag of this word is PRP in the Boxer's POS tagging, then it is decided as the final pronoun to be resolved.

Now we have 3 different decisions from above 3 modules, which need to be combined into one for finding the pronoun. In this process we give preference to the parsers, i.e. if the two parsers have produced the same word as the pronoun to be resolved, then it is the final pronoun to be resolved. Else string matching algorithm has preference if it has produced an answer, if not then the boxer's prediction is considered or else the Stanford parser's prediction is considered.

A consolidated example of pronoun resolution using above mentioned three approaches is given below.

Example 12 *Sentence : Frank was upset with Tom because the toaster he had bought from him did not work.*
question : Who had bought the toaster ?

The event maps for both the question and the answer are shown in examples in the previous section.

Boxer :

```
The entity to be resolved = person?
event e = buy
From the sentence event map :
agent of buy = person = male (indexed as 'he' in the XML output)
POS tag : PRP
Pronoun to be resolved = he
```

Stanford Parser :

```
entity to be resolved = who
event e = bought
From the sentence event map : agent of bought = who = he
POS tag : PRP
Pronoun to be resolved = he
```

String matching :

```
Words that match from question and answer : had bought
Since the whole string $had bought the toaster$ from the question
doesn't match with anything in the sentence, decision = null.

Final answer for pronoun to be resolved =  he
```

Chapter 3

Question Answer System

3.1 Use of Real World Knowledge

As discussed in the introduction, the winograd schema sentences contain some situations which involve applying real world knowledge to resolve the target pronoun in the sentence. This real world knowledge is obvious to the adults whose first language is English but not to the machine.

Another feature of winograd schema, as we have seen previously, is that the sentence can be about anything. Thus its not domain specific, which makes it harder to find answer for a machine. This is a very prominent feature of winograd schema that, it requires 'thinking' to get the correct answer with high probability and to do better than simply guessing, the system needs to understand what is going on. The use of real world knowledge is of utmost importance for this part of the system. When humans try to answer the question from the winograd schema, they don't merely guess the answer but they resolve the situation by using their background knowledge, which is not expressed in the sentence, but obtained from experience. To make the resolution harder the winograd schema challenge makers have used proper names, in the sentences, instead of using roles of the candidate answers. This requires us to resolve the proper names in the sentence to their semantic roles, which again requires real world knowledge of the situation presented in the sentence. It is worth mentioning that there are some recent advancements in the domain of world knowledge extraction, e.g. YAGO (Bryl et al. (2010)), Free-base (Lee et al. (2011)), DBpedia (Auer,Bizer et al.), but primarily the relations available in these are of the type 'IS-A' relations, e.g.: Salmon IS-A fish, which is not of much use to us while resolving the pronoun.

Here is an example of how the real world knowledge is required to resolve the situations in winograd schema sentences to find out the correct answer.

Example 13 *The trophy doesn't fit in the brown suitcase because its too small.*
What is too small?

- *Trophy*
- *Suitcase*

In the above example humans use background knowledge that smaller item fits into a bigger item and suitcase is a container for things in which the trophy will fit or won't fit. In our system we use commonsense knowledge-bases like ConceptNet or search engines in order to try to obtain background knowledge like above.

In our proposed system we mainly make use of the real world knowledge from the following five sources :

- *FrameNet* - A machine readable lexical database, based on the annotated examples of use of words and associated roles. Its main use in our project is for semantic role labeling.
- *Narrative Chains* - They are partially ordered sets of events centered around one common actor or protagonist. They are basically sequence of events with the syntactic role of the protagonist specified. We use narrative chains schema to explore knowledge about the the sequence of events in the winograd sentences.
- *ConceptNet* - Its a semantic network consisting of concepts and relations between them, that computers can use as a commonsense knowledge. ConceptNet contains every-day basic knowledge. It is used in our project as a one of the real world knowledge source to make associations between the candidate entities and the events or special words.
- *Google/Bing search* - As we know, the search engines are the biggest available source of real world knowledge. To exploit these sources evenly we use two search engines Google and Bing. Use of two search engines evens out any search engine specific irregularities in the results. We use search engines mainly to obtain the result count estimates for the semantic queries we form from the winograd sentences.
- *Sentiment Analyser* - Many of the Winograd sentences contain events which express positive, negative or neutral sentiment. If some these events are associated with the candidate answers and some with the target pronoun, then we can assign "goodness" or "badness" to these characters. And using that we match the target pronoun to candidate answer.

3.2 Measuring similarity between sentences and words

In our project we needed to take into account the semantic similarity or relatedness, between either the two sentences or the two words, at many places. We have used various semantic similarity mechanisms for this purpose, they are described in the following sections. Some of these measures use WordNet as their primary knowledgebase. Its a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept.

Sentence similarity

We use semantic similarity mechanism for sentences, to judge similarity in their syntactic behaviour. We use 'The similarity library' ($Pirro,\ Euzenat\ ISWC2010$) for measuring the sentence similarity. This library is an extension of the JWSL(Java Wordnet Similarity Library). This library makes use of the maximum weighted bipartite graph matching problem to match the words from two sentences.It uses the Hungarian method for the bipartite graph matching problem. This approach basically finds the best coupling for a word from one sentence to the word in other sentence. To find out the best coupling for a word it makes use of the two mechanisms described below i.e. using WordNet and using Search engine. We assume here that if the two sentences are 100% similar then their syntactic composition would be perfectly identical.

Sentence similarity Using WordNet

This category of similarity measure using the similarity library, exploits WordNet ontology. Also one point to note is that, they use POS(Part Of Speech) matching strategy for bipartite graph matching problem. Also this matching method uses the FaITH measure ($Pirro,\ Euzenat\ ISWC2010$) to compute the similarity and the relatedness between the two words. This measure makes use of the metric $msca(c1, c2)$ which represents the information content shared by the two concepts $c1$ and $c2$ in the WordNet ontology structure. An example of this technique is :

Example 14 *sentence 1 : Her letter upset me very much*

sentence 2 : Frank was upset with Tom because the toaster he had bought from him did not work

Similarity using WordNet mechanism of "the similarity library" : 0.7517095725863563

Sentence similarity Using search engine

This category uses the Google search engine to find similarity measure between the two words in the graph matching problem. The semantic relatedness measure, the library uses here, is called NGD (Normalized Google Distance). It's a statistic based approach for similarity measurement. It is computed by considering the number of hits, for a set of keywords, returned by the Google search engine. The concept behind this approach is that the similar meaning words will have lesser Google distance while unrelated words will be distant from each other in terms of Google distance. Although we use this mechanism, it has limitations that it can not be used for the sentences containing more than 4 words, hence we mainly use it to compare word definitions. An example of similarity using search engine is :

Example 15 *sentence 1 : was upset*
sentence 2 : disappointed

Similarity using search engine mechanism of "the similarity library" : 0.817477257634680

Word similarity

We use similarity measure for the words primarily to categorize them into groups. So if the two words are $x\%$ similar then we infer that there is a $x\%$ probability that these two words fall into the same category. Such categorization of words, based on the semantic relatedness, is mainly important in categorizing their syntactic behaviour. Thus we can estimate that $x\%$ semantically similar words will have atleast $x\%$ similar syntactic behaviour. For example we can replace two synonyms for each other in the sentence, in other words since they are semantically 100% similar their syntactic behaviour is perfectly identical in the sentences.

Word similarity Using WS4J library

Here we make use of the WS4J(WordNet Similarity for Java) library which uses WordNet to measure the similarity between two concepts. It provides several similarity algorithms, but we only use

WUP(Wu & Palmer, 1994) similarity measure. This measure calculates relatedness by considering the depths of the two synsets in the WordNet taxonomies, along with the depth of the LCS. The wordNet synsets are defined as "words that denote the same concept and are interchangeable in many contexts – grouped into unordered sets". The formula used by WUP measure to calculate semantic relatedness between words is :

```
score = 2*depth(lcs) / (depth(s1) + depth(s2))
```

Thus the score will be between 0 and 1(included). If the two words are same then the score will be 1. For example word similarity between words 'upset' and 'hurt' is 0.632, by using this method.

Word similarity Using ConceptNet

As compared to WordNet, the ConceptNet contains more informal and defeasible knowledge that is practical in nature and of everyday use. Hence we also use ConceptNet to calculate similarity between two concepts(words). As we have seen in the previous section, the ConceptNet is a large semantic network of concepts with relations as edges, connecting the concepts. Summing up the weights on different edges between concepts yields a large sparse matrix. They use dimensionality reduction techniques on this matrix and then calculate similarity between concepts by 'spreading activation' method. The ConceptNet API returns similarity weight between 0 to 1. 1 indicating the highest similarity between two concepts. For the same 'upset', 'hurt' example above the similarity count returned by ConceptNet is : 0.4609041242120129.

3.3 Pronoun Resolver System Architecture

An overview of our Pronoun resolver system architecture is shown in the figure 3.1 below.

In the upcoming sections, we will elaborate on each of the system component and its use in detail.

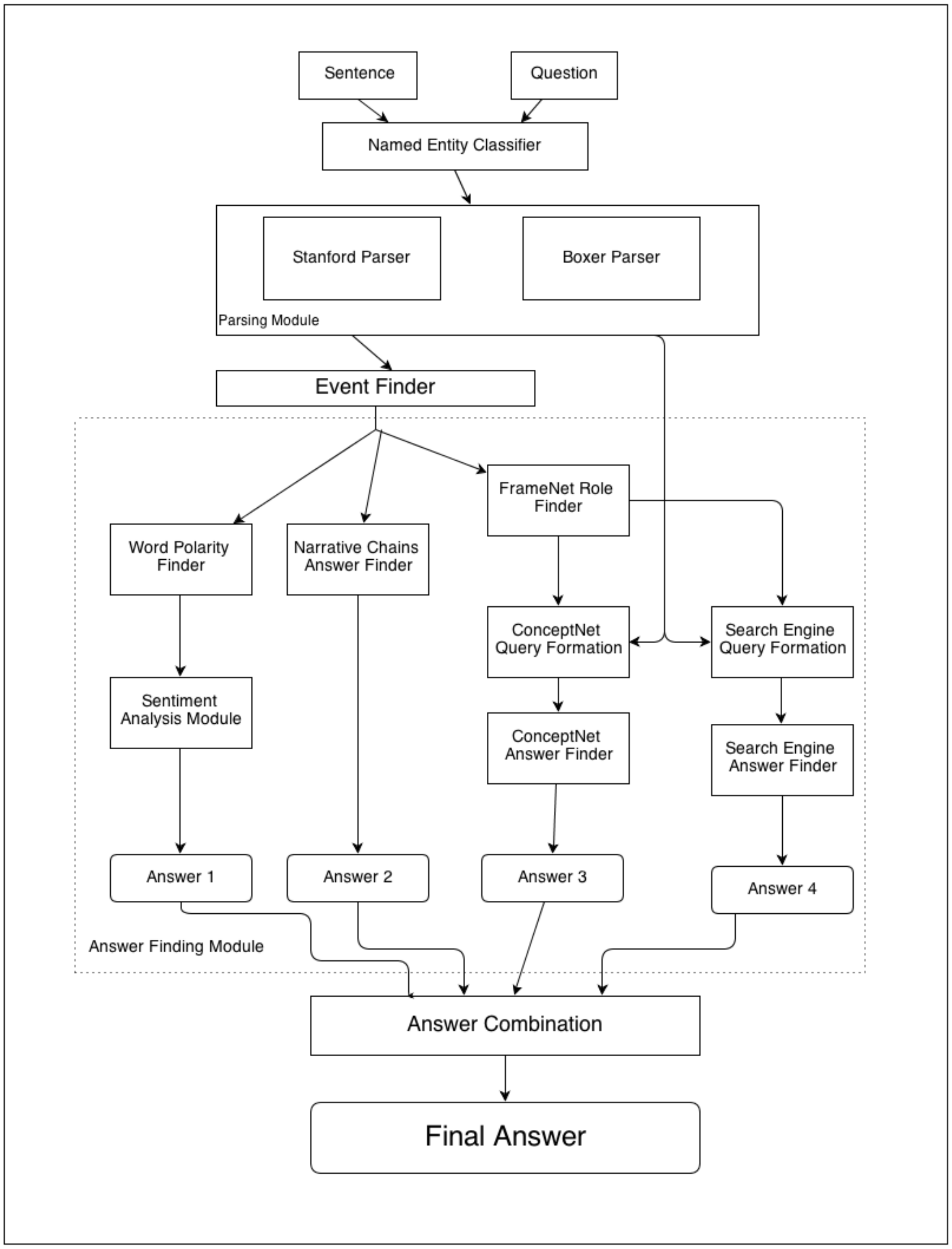


Figure 3.1: Overview of Pronoun Resolver system

3.4 Named Entity Classifier

We use a classifier in our research to filter out some of the Winograd sentences from the dataset, based on certain criteria. There is a web-page $''www.behindthename.com''$. This web page contains all the information about the person names. We use this web page's web-API to filter out sentences containing person names. The reasons behind using classification mechanism are explained in the below paragraph.

As explained in the section 1.2, Winograd schema makers try to avoid the pitfall of the resolver making associations between entity's semantic roles with certain words in the sentence or question, to find the answer, by using the person names in the sentence. This can be better shown with a small example :

Example 16 *Mother gave birth to the daughter, she is a beautiful baby.*
Who is a beautiful baby ?

In this example it is easy to make association of daughter with the baby instead of mother with the baby. Hence the target pronoun in this sentence, 'she', can be easily resolved to 'daughter' here. But to avoid this pitfall, Winograd schema makers present this sentence as :

Example 17 *Mary gave birth to little Ann, she is a beautiful baby.*
Who is a beautiful baby ?

So now the person or the automated system trying to resolve the pronoun 'she', would have to detect the semantic roles of $Mary$ and Ann, which are $mother$ and $daughter$ respectively, and then make the association that the daughter is a baby and hence Ann is the answer. This task requires deep parsing and very sophisticated semantic role labelling. The parsers used in this research, Stanford parser and Boxer both are shallow parsers and also the FrameNet semantic role labeller that we have designed doesn't always come up with the meaningful roles for both the entities in the sentence. For example,

Example 18 *Frank was upset with Tom because the toaster he had bought from him didn't work. Who had bought the toaster ?*

In this example, there is only one role associated with the event 'upset' i.e. 'Experiencer' and it is assigned to the subject of event 'upset' i.e. Frank. But Tom's role in this sentence remains unresolved. Hence we can't make associations based on roles. In this research, two out of our four modules(Search engine and ConceptNet) require semantic roles because they try to resolve the target pronoun by making associations of entity roles with different events or special words in the sentences.

Because of the above mentioned reasons our classifier rejects the Winograd sentence, if it contains person names. That is the sentence is not tried in our system for pronoun resolution.

3.5 Use of FrameNet

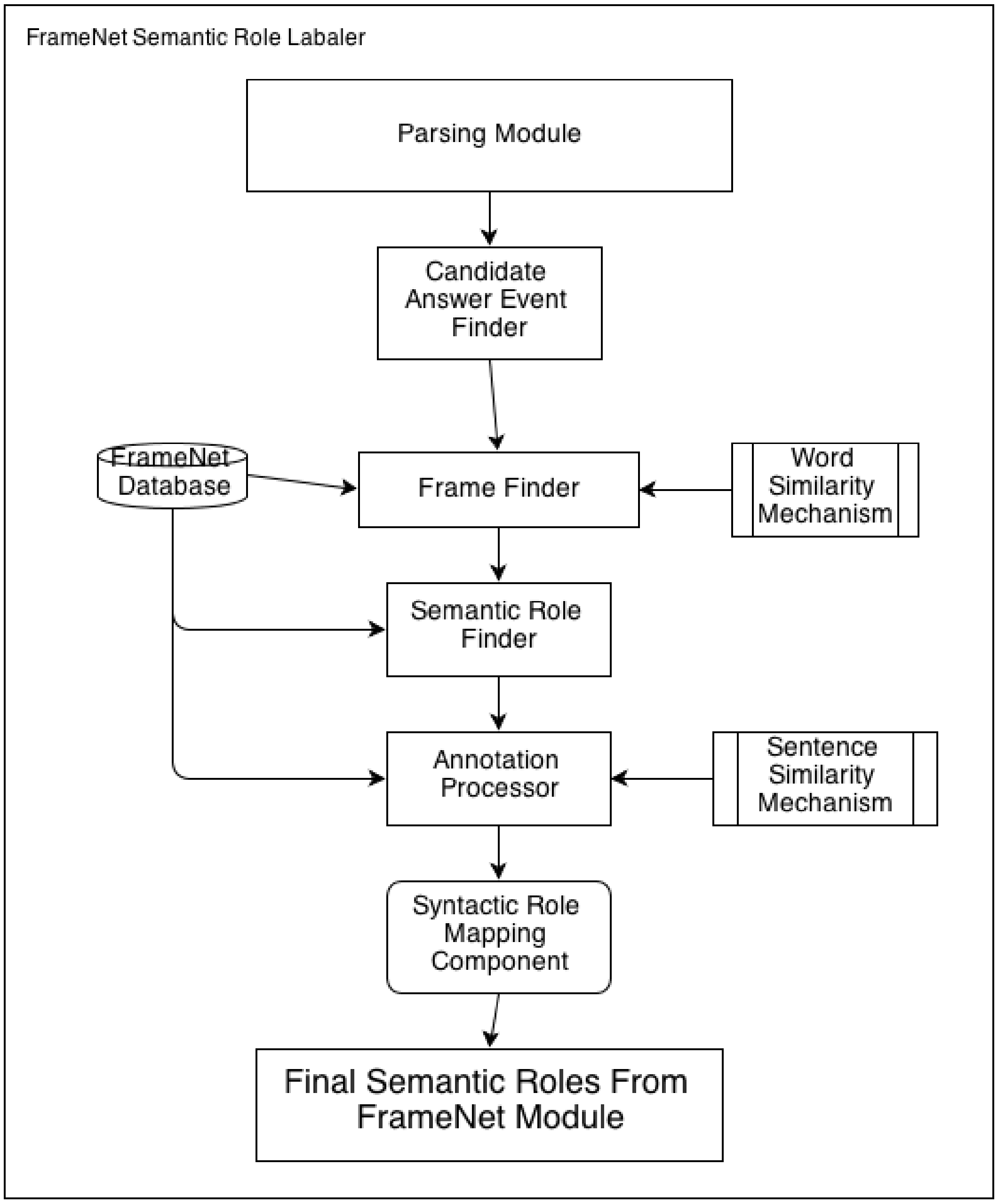

FrameNet Semantic Role Labaler
Parsing Module
Candidate Answer Event Finder
FrameNet Database
Frame Finder
Word Similarity Mechanism
Semantic Role Finder
Annotation Processor
Sentence Similarity Mechanism
Syntactic Role Mapping Component
Final Semantic Roles From FrameNet Module

Figure 3.2: Overview of the FrameNet Component

What is FrameNet ?

FrameNet is a lexical database. It is both human and machine readable. It provides us more than 170,000 manually annotated sentences which can constitute as a training dataset for semantic role labelling. FrameNet is based on the Frame Semantics (Fillmore and Baker et al., 2001). An event in the real world is represented as a semantic frame in the FrameNet. Each frame contains Frame Elements(FEs) which are description of the type of event, the participants in the event, stimulus leading to the event etc. There are certain events or words that invoke certain type of frames.

Example 19 *An event 'ask'(verb) will invoke two frames "Request" and "Questioning".*

FrameNet provides definitions for all the frames and gives us annotated sentences with respect to each of the frame. The sentences show us how the Frame Elements fit around the word that invokes this frame. The example of the annotated sentence for event 'ask' corresponding to the frame 'questioning' is :

Example 20 ($she_{speaker}$**)** *didn't have the nerve to ASK [*$him_{addressee}$*] [*$straight\ out_{manner}$*] .*

Here speaker, addressee and manner all are elements corresponding to the frame 'questioning'. There are 108 such manually annotated sentences provided in the FrameNet which correspond to this particular event for this frame. These sentences will give us a fair idea of how the event 'ask' is used in the various situations and what are the various participants in it when the context is 'questioning'.

The FEs are divided into core elements and non-core elements. Core elements are the important ones in invoking this particular frame. Each of the FEs have a semantic type associated with them, and there are about 40 semantic types defined in the FrameNet. The semantic types help in identifying the type of semantic role this particular FE plays when this frame is invoked. For example : in the above example for the event 'ask' corresponding to the frame 'questioning', the Frame Elements are speaker, addressee. Both of them have semantic type $sentient$ associated with them.

Why FrameNet ?

To avoid making an answer obvious to the user, the winograd schema makers try to use proper names, instead of using roles or entity types, in the sentence. This makes it more difficult for a machine to guess the correct answer, because if it is trying to establish the closeness between the candidate answers and a keyword in the sentence/question using large corpus based methods then it has to identify the semantic entity type/role in the sentence first, since the knowledgebases or the corpuses won't provide any closeness measure between a proper name and a keyword.

Example 21 *The women stopped taking pills because they were pregnant. Which were pregnant?*

Ans 0 : Women

Ans 1 : Pills

In this example it is easy to figure out, using knowledgebase or a search engine that women have closer association to the concept 'pregnant' than the pills. Hence it would be easy to figure out the answer. Taking this pitfall into account, the winograd schema makers replace the entity woman in the above sentence with some proper name such as Joan, susan etc. Thus in order to find the answer, the system has to first identify that the named entity in the sentence is of the type 'person' or 'woman' in particular.

Just as mentioned in the above example there can be two entities in the sentence which are mentioned by their proper names. In that case to make use of commonsense knowledgebases, as proposed in our approach, we need to resolve the proper names to their semantic roles in the sentence. This is where FrameNet plays very important role. FrameNet has various FEs related to the frames with which the event in the sentence associates itself. FrameNet also provides the manually annotated sentences, as mentioned in the previous section, which show us various ways in which the event and the corresponding FEs co-ordinate in the sentence.

How do we use FrameNet

We mainly use FrameNet to resolve the proper names in the winograd sentences to their semantic roles. We use events in the sentence, in which one or both of the candidate answers participate. The event maps which we prepare in the sentence parsing stage help us in identifying which candidates participate in which event and how. The use of FrameNet in our project is explained as below :

- Identifying the events - We use events in a sentence in which one or both of the candidate answers participate, as either subject/agent or object/patient. The events can be obtained from the event maps for the sentence which we prepare in the sentence parsing stage. We list all such events in the sentence and note which candidate participates in that particular event and how (as a subject or object).

- Search FrameNet - For each event e in the above mentioned event list, we search FrameNet for the correct frame corresponding to the context of event e in the sentence and search that frame for the semantic roles, that can be assigned to the subject/object of the current event e. FrameNet search works as follows :

  1. First we take the event and its use – POS(Part Of Speech) tag – in the sentence ($verb$/$adjective$/$noun$) and go to FrameNet to search for all the frames for the current pair (event e,use). For each such (frame,event,use) combination there is a Lexical Unit (LU) associated in the FrameNet. We store all such LUs found in a list. Now our task is to identify the correct frame (and LU) from amongst these several LUs corresponding to the current event used in the current context in a sentence. In the case of event not being found in the FrameNet repository, we make use of WordNet utility called Java Api for WordNet Searching (JAWS) to find the synonymous word with the same usage as the current event e, make it a new e and continue our FrameNet search with this event. For example : for the pair (ask,verb) there are 2 frames in the FrameNet : "Questioning" or "Request".

  2. Each of the LU, corresponding to the frame, has its own definition associated with it. We try to take each LU's definition, measure it against current event's context in the sentence and take the closest one as the LU to be considered for next stages. For considering context of event in the sentence, we combine all the verb forms surrounding the event along with the event itself. For example : Each of the two frames found in the above example, have definitions with respect to the event 'ask'. The definitions are as follows.

     ```
     event : ask
     ```

```
Frame : Request
    Definition : request to do or give something
Frame : Questioning
    Definition : say something in order to obtain an answer
    or some information
```

3. For matching the definitions of LUs with current context of event, we make use of the sentence similarity mechanism mentioned in the section 3.2. Out of the two mechanisms mentioned there we make use of the 'Search engine based similarity' mechanism preferably, however if the definition or the even context has more than 4 words then we use the fall-back mechanism of wordNet similarity. In some cases one LU might have multiple definitions associated with it, in such a case we find similarity between all the definitions and current context but take only the highest one for further consideration. In this stage we evaluate all such similarities of the current context of event with each LU's definition and we take the most similar one i.e. the one with the highest similarity to be the (frame,LU) combination for current event.
Continuing with the 'ask' event example, this task will try to find the similarity between each of above definition and the event's context in the sentence.

```
event : ask


Frame : Request
    similarity(ask,request to do) - 0.961664559273954
    similarity(ask,give something) - 0.7056825631086582
Frame : Questioning
    similarity(ask,say something in order to obtain an answer) -
    3.7084468364759324E-4
    similarity(ask,some information) - 0.8806850332722259


Based on the numbers above, highest similarity frame selected :
```

```
Request
```

4. We go to the frame found in the previous iteration in search of the semantic roles associated with it. As mentioned previously, each frame has some Frame Elements (FEs) associated with it. We take all the FEs for the current frame and search for the $core$ FEs of the semantic type "Sentient". These FEs are the actual semantic roles of the participating candidates. The role assignment step is explained in the next section.

The complete example explaining the above steps in actions is as follows :

Example 22 *Ann asked Mary what time the library closes, because she had forgotten . Boxer parse :*

```
event : ask
subject : Ann
object : Mary
```

1. *we take pair (ask,v) and go to FrameNet to find frames. There are 2 frames found - Questioning, Request.*
2. *Now we compare use of ask i.e. asked with the definitions of each of these frames. The result is as follows :*

   ```
   Processing Frame : Request
   Finding similarity :asked , request to do : 0.961664559273954
   Finding similarity :asked , give something : 0.7056825631086582
   Processing Frame : Questioning
   Finding similarity :asked , say something in order to obtain answer :
    3.7084468364759324E-4
   Finding similarity :asked , some information : 0.8806850332722259
   ```

   So we select 'Request' frame based on highest similariy = 0.961.
3. *After going to that frame we find that there are two semantic roles "Speaker" and "Addressee" of the type 'sentient'. We add them to the list of semantic roles found.*

4. *Also we note the LU associated with the triplet (ask,verb,Questioning).*

One thing to note here is that for searching in the FrameNet, we first have to know whether the candidate answer that is present in the sentence is a proper name or not. If its not a proper name then there is no need to resolve that. Currently while running the system we tell it which candidate answers are needed to be resolved to their roles based on the web API we use for classification purpose.

- Assign roles found to the candidates - This is a part where we try to assign semantic roles found in previous stage to the candidate answers present in the Winograd sentence that we are processing. This phase works in two main parts, one where the semantic roles found in the previous stage get assigned to the syntactic roles in the sentence and second where the candidates are mapped to the syntactic roles and by transitivity to the semantic roles found in the FrameNet. The inputs to this phase from the previous is the Lexical Unit (LU) that was found for the current event e. The assignment process works as follows :

  1. The LU contains manually annotated sentences with annotations mentioning the roles from the frame. We take each sentence and measure its semantic similarity with our main sentence using the WordNet based sentence similarity measure as explained in the section 3.2. Since the sentences are large we only use WordNet based similarity measure here. For example : The "Request" frame for 'ask' example contains 6 annotated sentences. The similarity measure finds and records similarity between the annotated sentences and current Winograd sentence as follows :

     ```
     Winograd Sentence :
     Ann asked Mary what time the library closes, but she had forgotten .


     Annotated Sentences :
     Their names were taken from Amal 's brother  but by the time they
     were collected and asked to appear before the police for
     statements they had taken great pains to prepare their story
     Similarity : 0.5276498622827088
     ```

```
One drop of water is not much to ask  but the king 's son had none
at all
Similarity : 0.5997520404154805

She expressed a warm opinion of the piece and asked for more of
her work
Similarity : 0.5826167405732824

One of the parents involved in the Orkney case actually asked that
the hearing should be held either that day or the next  to comply
with this section of the Act
Similarity : 0.5989547662776972

He took it down when asked to do so
Similarity : 1.0

In all the months that she 'd been going out with Adrian  he 'd never
once asked to talk to her like that  in that special way  during
school time
Similarity : 0.5484133702418276
```

2. We have a similarity threshold of 50% here, i.e. we only consider sentences that are 50% or more similar to the current sentence. We follow sampling approach henceforth for identifying the syntactic role to semantic role mapping. Under this approach we consider all the sentences that are above similarity threshold. For each such sentence we consider its annotation, i.e. the syntactic role/s to semantic role/s mapping. This is considered as weighted sample representing a particular syntactic role mapping to semantic role. The syntactic roles tagged in FrameNet are NP_EXT and NP_OBJ and some others. The weight of the sample is equivalent to the similarity of the sen-

tence to the current sentence. Continuing with the above example, we select all the sentences from the annotations, because all of them are above the desired threshold.

3. We collect all such samples, which are above similarity threshold, from the previous iteration. Then we decide which semantic role gets assigned to which syntactic role based on the weight of samples. The syntactic role with highest weight count wins and gets assigned to the corresponding semantic role. This process takes place for all the semantic roles that were found in the previous subsection.

4. One thing to note here is that in a case when there is no sentence above similarity threshold, we just decide semantic role to syntactic role mapping based on the 'valence syntactic pattern' count given for the annotated sentences in this LU. This count denotes the count of different patterns of semantic role to syntactic role mappings over all the manually annotated sentences in this LU.

5. Then we come back to the original problem that is to assign the candidates to the semantic roles from the FrameNet. As mentioned in the previous section, from the event maps we formed during the parsing stage, we have a mapping of which candidate participates in the current event e and how(subject/object). Also from the previous bullet we have a semantic role corresponding to this event and we know its syntactic role i.e. whether its a subject or object of current event.

6. The syntactic roles that can be returned from FrameNet can be NP_EXT or NP_OBJ or some others. We just consider NP_EXT or NP_OBJ for our purpose here. We interpret them as subject or object respectively. Then we check if the role of candidate in the current event is the same as syntactic role returned from the FrameNet. If the answer is yes then we map candidate to this semantic role from FrameNet.

7. This process repeats for all the events in the current sentence and tries to assign semantic roles to the candidates.

8. One thing to note here is that, we have made one assumption here that every sentence will have only two candidate answers in it. Since this assumption is made, at the end of the whole assignment process if one of the candidate proper name gets assigned to

the semantic role and other is not, then we assign it to the second semantic role that was found.

Consolidating on the example of sentence from previous subsection :

Example 23 *Ann asked Mary what time the library closes, because she had forgotten .*

1. *From the previous iteration we have found the LU associated with the current event ask, we go to the LU and process sentences that are above similarity threshold with the above sentence.*
2. *From processing these sentences we get following syntactic role counts for the semantic roles*

```
'Speaker' and 'Addressee' :
Role :Speaker
Syntactic role :NP_Ext   Confidence : 6.055857539202291
Syntactic role :CNI_   Confidence : 0.5038031067476518
Role :Addressee
Syntactic role :DNI_   Confidence : 5.018252174188793
Syntactic role :NP_Ext   Confidence : 0.5038031067476518
Syntactic role :NP_Obj   Confidence : 1.0376053650134995
Syntactic role :Sfin_Dep   Confidence : 0.0
```

3. *Thus the semantic role to syntactic role assignment is :*

```
Speaker NP_Ext
Addressee DNI_
```

4. *In the example, Ann is subject of event ask. Hence Ann's semantic role is detected as Speaker. Now since Mary is the object of event ask and there is one more role detected with event ask and it is unassigned, we'll assign it to Mary. Hence Mary's semantic role is Addressee.*

The end output of the whole above process is that we have got the proper names in the sentence resolved to their semantic roles with the help of FrameNet. One point to note here that if the syntactic roles detected for both of the candidates are something other than NP_EXT or NP_OBJ then this FrameNet module won't return any semantic role assignment. This is because other roles don't definitively bind to any of either subject or object. In that case we keep the proper names unresolved.

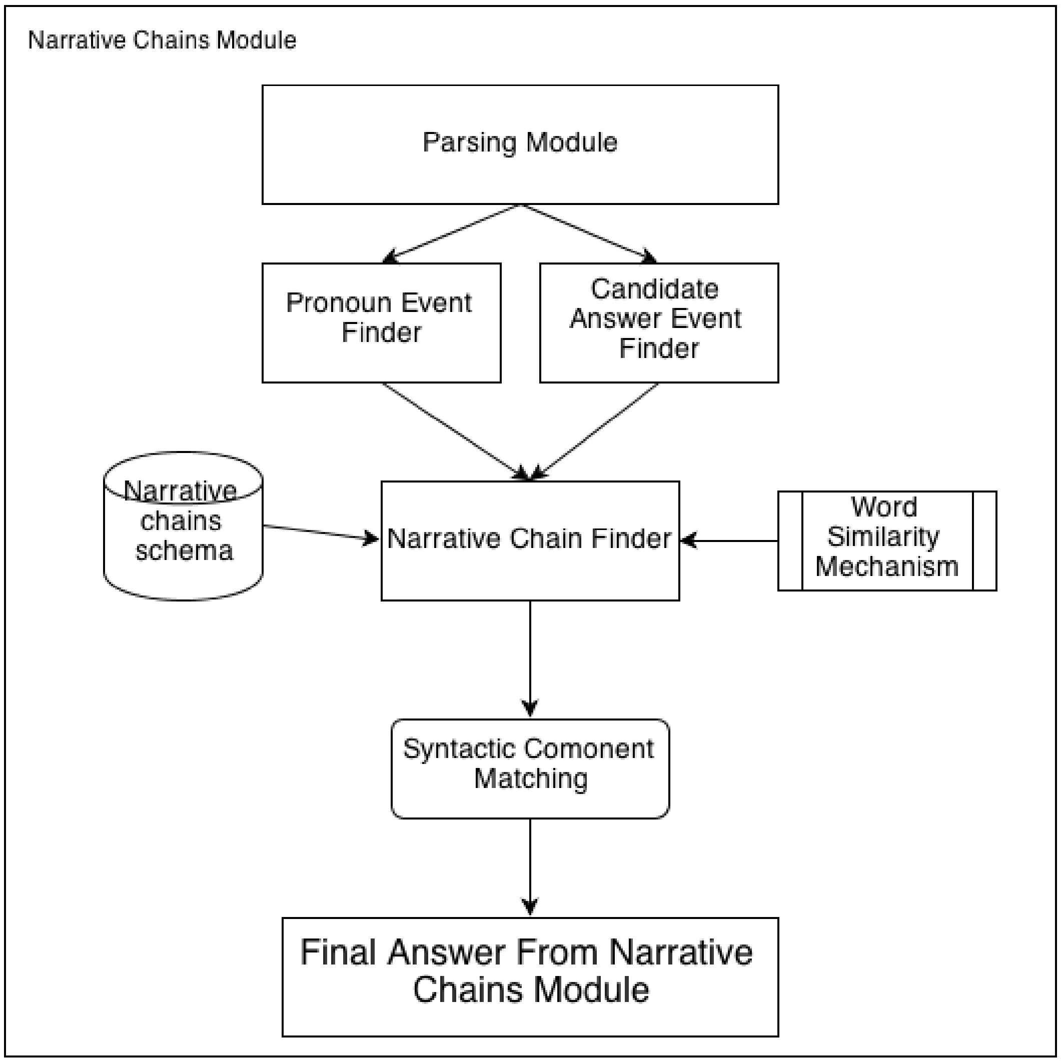

Narrative Chains Module
Parsing Module
Pronoun Event Finder
Candidate Answer Event Finder
Narrative chains schema
Narrative Chain Finder
Word Similarity Mechanism
Syntactic Comonent Matching
Final Answer From Narrative Chains Module

Figure 3.3: Overview of the Narrative Chains Component

Many a times in the Winograd sentence there is a sequence of events, with one event leading to another. The candidate answers and the pronoun to be resolved participate in these events. our challenge is to find the candidate answer to which the pronoun resolves to. Hence in order to find that, we need some ordering of events from the sentence and we need some knowledge about their(candidates and pronoun's) participation in these events.

Example 24 *Ann asked Mary what time the library closes, because she had forgotten .*

The reasoning we humans use while resolving pronoun is, somebody who forgets something asks someone else about it. Hence the target pronoun 'she' would be resolved to Ann. This reasoning is result of the knowledge that we humans acquire from the experience.

Our aim is to automate this kind of reasoning process. For that we will need some knowledge chain about how one event takes place after another and which actors participate in that chain. This kind of world knowledge can be inferred from Narrative Chains(NCs). They provide us with the story containing event based description of the participation of a common central actor called the protagonist.

What is Narrative Chains Schema ?

Narrative chains are partially ordered sets of events centered around a common protagonist, aiming to encode the kind of knowledge provided by scripts ($Schank\ and\ Abelson,\ 1977$). They can be learned from the unannotated text. They are a sequence of events, in a story, with the role of the protagonist or the actor denoted as $-s\ :\ subject$ or $-o\ :\ object$. Example of a narrative chain is :

Example 25 `remove-s cut-s dry-s peel-s slice-s`

This story tells us that the protagonist or the actor who removes something then cuts something then dries that something then peels and slices that something.
In more formal words, the protagonist does actions remove, cut, dry, peel, slice one after other or he is the subject of all these actions.

One thing to note here is that the Narrative chains are partially ordered that is there is a focus on the 'before' relation between the events but the event $e1$ that occurs before $e2$ is not a precondition of to the event $e2$. The partial ordering between these two events is just a common likely ordering in the spoken or written world. The schemas are clusters of verbs and their actors i.e. protagonists, the clusters are grown until they reach the maximum specified size. For the purpose of our experiment we use narrative chains schema of size 12 that are already extracted by Chambers and Jurafsky

(2008). This means that the schema we use contains narrative chains of size upto 12 i.e. one such chain as shown above, can contain maximum 12 verbs in the order.

Use of Narrative Chains in our Project

As described above, the narrative chains schema helps us in understanding event semantics in the sentence. We make use of its partial ordering property and the common protagonist in trying to prdict the answer for the Winograd sentence. As we know, in the Winograd sentence, there is a target pronoun, p to be resolved and two candidate answers $c1$ and $c2$. Also in almost all of the Winograd schema sentences there are two sentences, first one with candidate answers participating in some event/s and then the second one with the pronoun participating in some other event, and these two sentences are connected via a discourse connective like 'but', 'because' etc. As described in section 2, from the sentence parsing stage, we have got event maps and the order in which these events happen in the sentence. Also we have a NC schema stored in a text file, with each line in the file being a narrative chain. With this in hand, we make use of the narrative chains schema in the following way :

1. We make a list l_c of all the events e_c in which the candidate answers $c1$ and $c2$ participate and how (i.e. as a subject or object). The list consists of strings of type $event - s\#candidate$, where -s denotes participation of the candidate as a subject in that event. If the candidate participates in the event as an object then the string would be $event - o\#candidate$. Consider following example -

   Example 26 *The drain is clogged with hair , it has to be cleaned .*
   The event maps for this sentence are :

```
candidate answers : drain,hair
pronoun to be resolved : it

Event    Agent    Patient Map :
clean    null     it
clog     null     drain
```

```
list of candidate events is :  { clog-o#drain }
```

2. We then make a list l_p of the events e_p in which the target pronoun participates and how. This is a list of strings as $event - s$ or $event - o$ depending upon whether the pronoun participates in the event as a subject or an object respectively.

   Example 27 *Continuing with the previous example of 'drain', 'hair',*

   ```
   list of pronoun events is : clean-o
   ```

3. With the above two lists extracted we do the following :

Algorithm 1 $Narrative\ Chains\ Schema\ answer\ finding\ algorithm$

```
for ($e_p \in l_p$) do
    for ($e_c \in l_c$) do
        $decisionConfidence = searchNc(event\ in\ e_c, e_p)$
        The decision returned will be either $s$ or $o$
        if (participation of candidate in event = decision) then
            $confidence[candidate] = confidence[candidate] + confidence\ returned$
        end if
    end for
end for
if ($confidence[candidate1] > confidence[candidate2]$) then
    $Return\ decision = candidate1\ with\ confidence = confidence[candidate1] - confidence[candidate2]$
else if ($confidence[candidate2] > confidence[candidate1]$) then
    $Return\ decision = candidate2\ with\ confidence = confidence[candidate2] - confidence[candidate1]$
else
    $Return\ no\ decision$
end if
```

Example 28 *For continuing example, we try to search for the narrative chains containing* $clog - o..., clean - o$.

4. At the end of an execution of the above algorithm, we'll have final confidence array consisting of confidence score on the two candidates $c1$ and $c2$. And we take the candidate with the greater confidence and pronounce it as the winner i.e. the answer to this question or the correct resolution of the pronoun in the sentence.

Example 29 *continuing our previous example, if there is a desired chain found say* $clog - o..., clean - o...$, *then we increase the confidence on the candidate 'drain' by 1, because the list contains 'clog-o#drain'.*
But as stated in the "searchNc" algorithm below, if both the pairs are not found, then we try to find the chains which contain $........, clean - o$. *Such chains are found, then we compare the similarity of the prior events in the chains to the event 'clog'.*
For this example, there are 3 such chains found which contain 'clean-o'.

```
The three chains are :

1>  rub-o smoke-o eat-o cover-o soak-o drink-o cook-o drain-o dry-o

clean-o

2>  paint-o sand-o stain-o finish-o strip-o cover-o refinish-o varnish-o

remove-o clean-o

3>  cook-o store-o pack-o dry-o check-o remove-o clean-o


Now we evaluate similarity of each of the event above, with 'clog' event.


We find that there are 2 events which are more similar than

the threshold of 0.5, These are

similarity(clog,cover) = 0.5937941451370026

similarity(clog,paint) = 0.563729650516104
```

```
Each of these have actor as '-o', hence we take average of similarity
count over these 2 events.
If there was some event with '-s', which was more than 50% similar,
then we would have taken average over them as well.

Hence the final decision is
-o has 0.5787618978265533  weight
-s has 0 weight

This is similar to having 0.578 *(clog-o,clean-o) pair with  1  weight.



Hence the final decision is clog-o with  0.578  weight.
Thus we conclude that the final answer is 'drain' since the list contains
'clog-o#drain'.
```

5. The $searchNc$ function called by the above algorithm looks like following :

Algorithm 2 $Searching\ algorithm\ for\ Narrative\ Chains\ Schema$

```
function SEARCHNC(String $candidateEvent$, String $pronounEventParticipation$)
    for ($each\ line\ l\ \in narrative\ chain\ schema$) do
        if ($l$ contains the pair $candidateEvent$ and $pronounEventParticipation$) then
            Extract candidate event and its participation in event ($s$-subject or $o$-object)
            Add candidate event and its participation to the list $pairs$.
        else if ($l$ contains just the event $pronounEventParticipation$) then
            Add all the events before this event in the line $l$ to the list $candidateList$
        else if ($l$ contains just the event $candidateEvent$) then
            Add all the events after this event in the line $l$ to the list $pronounList$.
            Also add protagonist in the $candidateEvent$ in $l$ at the top of current element in the list $pronounList$.
        end if
    end for
    if (List of $pairs$ is not empty) then
        Extract the participation of candidate in the event
        for (Each unique event $p$ in the $pairs$) do
            if (Participation is $s$ i.e. subject) then
                Increase confidence on decision as subject by 1.
            else if (Participation is $o$ i.e. object) then
                Increase confidence on decision as object by 1.
            end if
        end for
    else if (List $candidateList$ not empty) then
        for (Each unique event $cl$ in the $candidateList$) do
            if (Similarity between $cl$ and $candidateEvent$ i.e. $similarityWeight$ > $Threshold$) then
                Consider how the protagonist participates in this event.
                if (Participation is $s$ i.e. subject) then
                    Increase confidence on decision as subject by $similarityWeight$.
                else if (Participation is $o$ i.e. object) then
                    Increase confidence on decision as object by $similarityWeight$.
                end if
            end if
        end for
```

Algorithm 3 $Searching\ algorithm\ for\ Narrative\ Chains\ Schema$ (continued)

```
        if (confidence matrix is empty i.e. no event from the above iteration was greater than the similarity threshold) then
            In this case we consider $pronounList$.
            for (Each unique event $pl \in pronounList$) do
                if (similarity between $pl$ and event $pronounEventParticipation$ i.e. $similarityWeight > Threshold$ & protagonist in $pl$ = protagonist in $pronounEventParticipation$) then
                    Extract protagonist participation for $candidateEvent$ at the top of the current element.
                    if (Participation is $-s$ i.e. subject) then
                        Increase confidence on decision as subject by $similarityWeight$.
                    else if (Participation is $-o$ i.e. object) then
                        Increase confidence on decision as object by $similarityWeight$.
                    end if
                end if
            end for
        end if
    end if
    Calculate average confidence on decisions as both subject and object.
    if (Average confidence on decision subject is greater) then
        Return decision = $s$ and $confidence = (confidence[subject] - confidence[object])/confidence[object]$
    else if (Average confidence on decision object is greater) then
        Return decision = $o$ and $confidence = (confidence[object] - confidence[subject])/confidence[subject]$
    else
        Return $null$
    end if
end function
```

Some points to note about the above mentioned process of finding the answer in NC schema :

- Motivation behind the use of similarity mechanism - Sometimes we don't find the narrative chain containing the two events we are looking for. In that case the pronoun will stay unresolved,and since narrative chains contain a limited set of distinct events, its efficiency to resolve the pronoun is hampered. As an example :

Example 30 *I spread the cloth on the table in order to display it .*

In the example, 'it' participates in the event 'display' as a subject according to Boxer output and 'cloth' participates in the event 'spread' as an object. Here spread is a candidate event and display is a pronoun event. So according to the algorithm, we form the pair $(spread, display - s)$. Then we try to extract all the narrative chains containing both the elements from the pair. If any such chain is extracted then we can decide the answer based on the protagonist's participation in spread event in that chain. But unfortunately in the current case, there is no such chain extracted. Hence we extract all the chains which contain just the $spread$ event or just the $display - s$ event. There is no chain in the schema containing $display - s$. But we find 4 chains which contain $spread$ event. The 4 chains are as follows :

```
1>  identify-o discover-o transmit-o kill-s infect-s contract-o carry-o
cause-s detect-o report-o found-o spread-s
2>  detect-s kill-o report-s found-s spread-o carry-s discover-s
transmit-s infect-o contract-s identify-s cause-o
3>  own-s produce-s divide-s proliferate-s multiply-s sell-s buy-s grow-s
build-s acquire-s spread-s form-s
4>  threaten-o disrupt-o fight-s kill-o spread-o destroy-o cause-s
burn-o force-o contain-s damage-o set-s
```

Now as mentioned above, we know that 'cloth', one of the candidate answers, participates in the $spread$ event as object. Hence chain 1 and 3 are of no use to us, since they contain $spread - s$ as an element in the chain, and from the parser output we don't have any candidate participating in the $spread$ event as a subject. The chains 2 and 4 are useful to us because they contain $spread - o$. The original pair we were searching for was $(spread, display - s)$ and $display - s$ is not present in the chain 2 or 4. But what if some synonym of $display$ event is present in the chain ? Since they both belong to the same semantic class, they will have a protagonist participating in them in the same way. Now we apply this logic in reverse, i.e. we know that there are events $carry, discover, transmit, contract,$ $identify$ in the chain 2 and events $cause, contain, set$ in the chain 4, where $spread - o$ is also present, we know that the protagonist also participates in all these events as $-s$, our job is to check whether any of these is a synonymous word to $display$. Hence we use similarity measurement mechanism to measure if any of the event is semantically similar to the event $display$. In fact there is one event $set - s$ which is 50% similar to the event $display - s$. Thus we get a pair $(spread - o, set - s)$ out of the 4th chain which can also be written as $0.5 * (spread - o, display - s)$. Hence we can say that, in this chain, there is 50% possibility that the protagonist in the chain is the object of $spread$ event and subject of $display$ event. From this extracted chain, we can obtain the role played by the pronoun(i.e. the protagonist), in the candidate event $spread$. The role is $-o$ with 0.5 probability. Thus we derive that the pronoun 'it' resolves to the candidate 'cloth', which plays $-o$ role in $spread$ event. But since we didn't find the original pair we were looking for, we decide that the confidence on this answer is equal to the similarity of the event set to event $display$ i.e. 50%.

- Use of similarity mechanism - We use two similarity mechanisms for function $searchNc$ to calculate $similarityWeight$ described in the above algorithm. $similarityWeight$ is the similarity between the two events that are being considered. The two similarity mechanisms are word similarity calculation using ConceptNet and ws4j library. Both of these mechanisms are described in detail in the 3.2 section. We send the events to the similarity calculation mechanism which calculates the two similarities and returns the average similarity. This way we try to avoid any imbalance or favouritism introduced by one of these two similarity calculation mechanism.

- Similarity threshold for similarity mechanism - We use a similarity threshold $Threshold = 0.5$ for the events to be qualified to make the decision. This way if one of the similarity mechanism produces a very high similarity count for a pair of events and the other mechanism produces a bit lower count such that their average is still higher than or equal to 0.5 then the event would be considered in the decision making process.

- Searching NC schema with event inputs from both parsers - As we have described in the section 2, we are using two parsers for parsing the Winograd sentence. Hence we get two different event maps for the same sentence and hence we perform the above mentioned procedure, for resolving the pronoun to the candidate, twice pertaining to the two input event maps.

- Resolving events to their base forms - As we have seen in the 2.2 section, the Stanford parser doesn't resolve the events in the sentence to their base forms. The base form of an event is the present tense form with respect to the verb or a singular form for the noun. So while using the above procedure with respect to Stanford parser, we have to take care to resolve both candidate answer events and the pronoun event/s to their base forms before passing them to the function $searchNc$. We make use of the Java Api for Wordnet Search(JAWS) utility, which allows us to use WordNet knowledgebase easily, for this purpose. Thus we use WordNet to get the base forms of the events that we get from the Stanford parser. This precaution of resolving the events to their base forms is necessary because narrative chains schema only contains the chains of base forms of events.
  There is one fix that we apply to the output from the JAWS library. Sometimes JAWS library doesn't return the correct base form for the verbs that end in ed, e.g. for the verb 'asked' it returns 'aske'. So we perform this validation that if the verb ends with ed and the returned base form ends with e then we remove that trailing e.

- combining the results and predicting the final answer - Finally the important question is how to combine the results we get from running the above procedure twice, once for each parser and predict the final answer. We follow following algorithm to get the final answer by combining the two sources. The algorithm basically gives out no decision if the outputs from two parsers are different. The algorithm produces decision if both parsers produce same output, or one of them produced no decision.

Algorithm 4 *Final answer prediction algorithm*

```
if Confidence matrix is null for Boxer then
    Return decision from Stanford parser
else if Confidence matrix is null for Stanford parsers then
    Return decision from Boxer
else
    Take average of confidence matrices returned by Boxer and Stanford parser.
    if ($avgconfidence[candidate1] > avgconfidence[candidate2]$) then
        Return decision = $candidate1$ with $confidence = avgconfidence[candidate1] - avgconfidence[candidate2]$
    else if ($avgconfidence[candidate2] > avgconfidence[candidate1]$) then
        Return decision = $candidate2$ with $confidence = avgconfidence[candidate2] - avgconfidence[candidate1]$
    else
        Return no decision
    end if
end if
```

The above mentioned process combined, can be shown in action with the working example as follows :

Example 31 *Ann asked Mary what time the library closes, because she had forgotten .*

- *Event maps from two parsers :*

```
Boxer :

Event   Agent   Patient Map :

ask     ann     time

forget  female


Stanford :

event       subject     object

asked       Ann         Mary

forgotten   she         null
```

- *From the above maps, these two parsers will predict the answer individually using NC schema.*

- *Using Boxer :*

```
Pronoun events :
forget-s
Candidate Answers Events :
ask-s#Ann

Function call : searchNc(ask,forget-s);
Output :
own-s raise-s report-s mention-s sell-s ignore-s neglect-s overlook-s
buy-s pay-s address-s forget-s
similarityWeight = 0.5736309186957134 for (ask,report)
similarityWeight = 0.6613875689700144 for (ask,ignore)
similarityWeight = 0.5424787399930289 for (ask,address)

head-s parry-s fight-s connect-s challenge-s meet-s win-s
inject-s thrust-s forget-s talk-s demonstrate-s
similarityWeight = 0.5447995830296357 for (ask,meet)

decision : s
confidence : 58.057420267209814 %

Boxer Answer = ask-s = Ann         confidence = 58.057420267209814 %
```

- *Using Stanford parser :*

```
Pronoun events :
forgotten-s
Candidate Answers Events :
asked-s#Ann
asked-o#Mary
```

```
base form(asked) = ask

base form(forgotten) = forget


Function call : searchNc(ask,forget-s);

Output :

own-s raise-s report-s mention-s sell-s ignore-s neglect-s

overlook-s buy-s pay-s address-s forget-s

$similarityWeight$ = 0.5736309186957134 for (ask,report)

$similarityWeight$ = 0.6613875689700144 for (ask,ignore)

$similarityWeight$ = 0.5424787399930289 for (ask,address)


head-s parry-s fight-s connect-s challenge-s meet-s win-s

inject-s thrust-s forget-s talk-s demonstrate-s

$similarityWeight$ = 0.5447995830296357 for (ask,meet)


decision : s

confidence : 58.057420267209814 %


Stanford Answer = ask-s = Ann    confidence = 58.057420267209814 %
```

- *Combining the results :*

```
Since both the parsers have produced an answer, we take average

of the confidence.

Final answer = Ann            confidence = 58.057420267209814 %
```

3.7 Use of ConceptNet

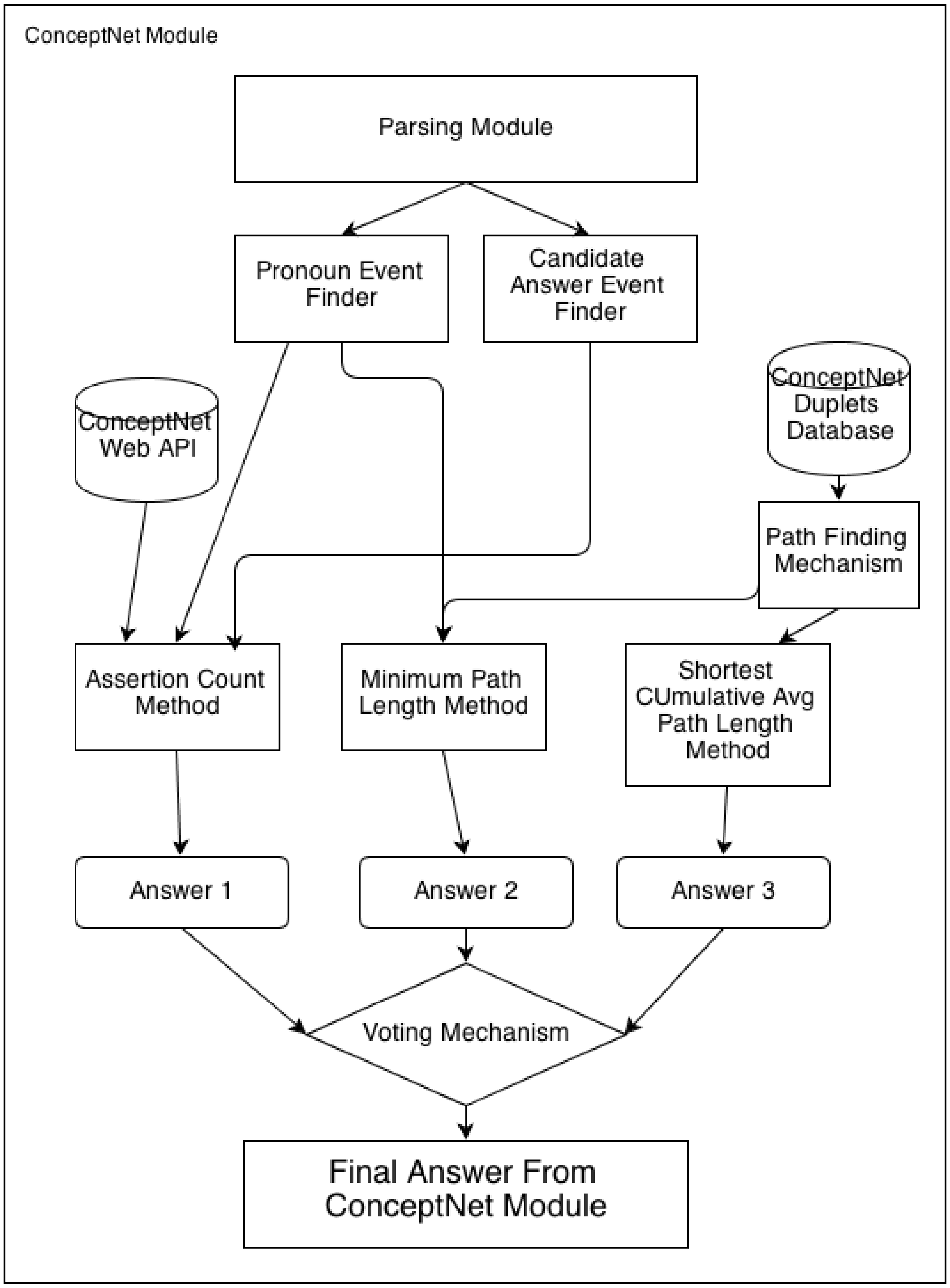

ConceptNet Module
Parsing Module
Pronoun Event Finder
Candidate Answer Event Finder
ConceptNet Web API
ConceptNet Duplets Database
Path Finding Mechanism
Assertion Count Method
Minimum Path Length Method
Shortest CUmulative Avg Path Length Method
Answer 1
Answer 2
Answer 3
Voting Mechanism
Final Answer From ConceptNet Module

Figure 3.4: Overview of the ConceptNet Component

What is ConceptNet ?

ConceptNet is a freely available commonsense knowledgebase. It's a semantic network of concepts, where nodes are concepts and the edges are the relations between them. The concept in a ConceptNet can be words or short phrases in the natural language. Main strength of this knowledgebase is that it contains concepts about everyday basic knowledge, cultural knowledge, scientific knowledge etc. ConceptNet can also be imagined as a hypergraph, i.e. it contains edges about the edges. Each statement in ConceptNet has some justification i.e. Starting from one source concept we can reach to some destination concept, by travelling via edges, and each of the edges or nodes we visit in this process will constitute to the justification of relationship between the two concepts. ConceptNet 5.1 has a web API. It offers a search feature, which can be used to find the number of edges that match a certain criteria. The result returned is in JSON format. They also make their knowledgebase available for download, in the JSON format, with the hypergraph flattened. The JSON consists of an array of ConceptNet edges. That is each edge in the downloaded jSON would contain a 'start' word, 'relation', 'end' word fields from which the graph could be reconstructed. Each of the 'start word' or the 'end word' is a natural language concept and the 'relation' is an edge connecting these two concepts in the graph.

How do we use ConceptNet

Use of KnowledgeBase

We use ConceptNet to draw inferences about the association between various concepts presented in the Winograd sentence and question. We use ConceptNet in two ways available, as mentioned above, one is through the web API and the second way is by downloading the knowledgebase, converting it to desirable format and using it via ASP(Answer Set Programming). The details are as follows :

1. For the first of the two methods mentioned above, we use web search API of conceptNet, that allows us to search for the list of edges that match a certain search criteria. The base url for search API is : $http://conceptnet5.media.mit.edu/data/5.1/search$. We can specify some additional parameters for searching. The parameters that we use for our research are

$start$ and end, they specify the start and end of an assertion. The search with these parameters returns the edges in the graph whose corresponding start and end fields are same as those mentioned in search criteria, with different relations between them. An example of search using web search API is as follows :

```
Search URL to find relations between 'axe' and 'cut'.
http://conceptnet5.media.mit.edu/data/5.1/search?start=/c/en/axe
&end=/c/en/cut


Result :
axe, CapableOf, cut_wood
axe, UsedFor, cut_wood
axe, UsedFor, cut


result count = 3
```

We use this count in the inference mechanism discussed later in this section.

We count the number of different relations that exist between two concepts. Since ConceptNet is a hyper-graph and there are several types of relationships that can exist between the two concepts, while searching for the relations between two concepts $C1$ and $C2$, we query ConceptNet web search API twice, with two different queries, once by using start=$C1$ and end=$C2$ and then again with start=$C2$ and end=$C1$, because a backward relationship might also exist between the two concepts. We have developed a java method $getAssertionCount(start, end)$ which will perform the above mentioned functionality and return the result count.

Although we count the number of different assertions, we don't account for the assertions involving IsA relationship, because such assertions in the ConceptNet hyper-graph contains a lot of noise and it is not reliable to draw inferences using these assertions.

2. For the second method i.e. for using local copy of the ConceptNet knowledgebase, we downloaded the knowledgebase in the JSON format. From that we extracted the edges

of the graph and converted them to the desired 'has()' format, which are duplets, for later use with the ASP. For example the relation $leg, UsedFor, run$ is converted to $has(leg, run)$, $has(run, leg)$. duplets. Each edge in the ConceptNet's JSON file will have $start$ and end field indicating the start concept and end concept of the assertion. We use these two fields to create two duplets from them, $has(start, end)$ and $has(end, start)$, indicating that there exists a relation between the two concepts : $start$ and end. Note that we only use core ConceptNet relations for preparing the duplets' database, that way we omit mostly the Db-pedia assertions and some other assertions containing location information, person specific information or some of the trivial IsA relations etc. For example : we omit relations such as $shaw\_cor, IsA, public\_company$ or $barack\_obama, IsA, president$. These are the assertions that are useful when we face an open domain knowledge question but not for the Winograd schema questions where we need more general kind of world knowledge from which we can draw inferences.

When preparing the duplets, we omit negative assertions,i.e. the assertions whose relation field starts with word 'not', from the ConceptNet – it is as if since they are negative relations they don't exist. We also omit assertions whose relation field indicate 'TranslationOf', for example we omit relations such as $Punaise, TranslationOf, darn$ indicating that the word 'punaise' is a french word for the word 'darn'. Also ASP doesn't allow us to use the word 'not', since it's an internal keyword, we convert 'not' occurring in $start$ or end to 'notx'. We also remove any noise(i.e. punctuation marks, calculation symbols and other special symbols), which are not an alphabet, from the 'has' duplet. The 'has' duplets are stored in the text files with each line containing one duplet. For example, duplets prepared from the relation $axe, UsedFor, cut$ are $has(axe, cut)$ and $has(cut, axe)$. The duplets database so formed contains more than 3 million duplets.

3. Since ConceptNet is a hypergaph, we can travel from one concept to the other concept via edges. We use of this idea while using the duplets database described above. Basically we can find path lengths between the source concept and the destination concept. We use ASP for this purpose because it's a logical programming language and it efficiently solves search problems. We have written a set of ASP rules to search for a path between two concepts in

the ConceptNet duplets database. We restrict the maximum path size for the search to 4, considering large number of duplets to be searched. Below are the ASP rules that we use to find path length :

```
%the path length is 1 if there is relation between any two words
path(U,V,1) :- has(U,V).


%path between X and Y via Z, of length T+1
%alternative 1
path(X,Y,T+1) :- alternative1_contains(X), path(X,Z,1),
path(Z,Y,T), T<=4, X!=Z, Z!=Y, Y!=X.
path(X,Y,T+1) :- alternative1_contains(X), path(X,Z,T),
path(Z,Y,1), T<=4, X!=Z, Z!=Y, Y!=X.


%alternative 2
path(X,Y,T+1) :- alternative2_contains(X), path(X,Z,1),
path(Z,Y,T), T<=4, X!=Z, Z!=Y, Y!=X.
path(X,Y,T+1) :- alternative2_contains(X), path(X,Z,T),
path(Z,Y,1), T<=4, X!=Z, Z!=Y, Y!=X.


%the two words are related if there is a path in either direction
%of path length <=4
%alternative 1
alternative1_relation(X,Y,T) :- path(X,Y,T), alternative1_contains(X),
premise_contains(Y).
%alternative 2
alternative2_relation(X,Y,T) :- path(X,Y,T), alternative2_contains(X),
premise_contains(Y).


#hide.
```

```
#show alternative1_relation(X,Y,T).

#show alternative2_relation(X,Y,T).
```

For example If we want to find paths between $(lawyer, ask)$ and $(witness, ask)$ then this can be input to the above set of rules as follows :

```
%premise
premise_contains(ask).
%alternative 1
alternative1_contains(lawyer).
%alternative 2
alternative2_contains(witness).
```

The answer is output as follows :

```
Answer: 1
alternative1_relation(lawyer,ask,5) alternative1_relation(lawyer,ask,4)
alternative1_relation(lawyer,ask,3) alternative2_relation(witness,ask,5)
alternative2_relation(witness,ask,4) alternative2_relation(witness,ask,3)
alternative2_relation(witness,ask,2)
SATISFIABLE
```

We use the above output to find the minimum path length between the two concepts. Using that data we can make some inferences. We have a Java method $getMinPath$, which takes as input lists of premise, alternative1,alternative2 and feeds them to the ASP solver using $clingo$. The method returns all the paths, of length less than or equal to 4, from each of the alternative1 & alternative2 to the premise.

Inference Mechanism

After describing how we use the ConceptNet knowledgebase in two different ways, we'll discuss how we use above information to draw inferences regarding Winograd sentences and questions to find out correct answer. We use three mechanisms to draw different inferences from the knowledgebase. At each of these three stages we use event maps and other information obtained from the sentence parsing stage, as described in section 2. The inference mechanisms used are as follows :

- *Counting Assertions* : Here we count the number of different assertions that exist for the two concepts, using web API, as described above. First we use the Boxer parser's output to count assertions. We obtain event-agent-patient map of the Winograd sentence from the sentence parsing phase. That way we obtain each event Ec in which the answer candidates participate. We also note how the candidate participates in the event, i.e. either as $-s : agent$ or $-o : patient$. Along with that we obtain the event Ep in which the target pronoun, participates and in which way : $-s$ or $-o$. With this we call method $getConfidenceAssertionMethodConceptNet$ which draws inference over candidates and returns the candidate answer along with the confidence count. The algorithm used in the $getConfidenceAssertionMethodConceptNet$ method is described below :

Algorithm 5 *Find correct candidate answer by Assertion count method*

```
function GETCONFIDENCEASSERTIONMETHODCONCEPTNET(String $pronounEvents$, String $candidateEvents$) Search ConceptNet for the event pair/s with maximum count :
    $maxCount = 0$
    for ($e_p \in pronounEvents$) do
        for ($e_c \in candidateEvents$) do
            $count = getAssertionCount(e_p, e_c)$, using ConceptNet web API.
            if ($count > maxCount$) then
                make a new list maxPairs with current pair $(e_p, e_c)$.
            else if ($count = maxCount$) then
                add current pair $(e_p, e_c)$ to the existing list maxPairs.
            end if
        end for
    end forProcess the list of max pairs
    $confidenceArr[2]$
    for ($e_l \in maxPairs$) do
        if ($e_p$ in current $e_l$ contains target pronoun participation as $-s$) then
            check agent of corresponding $e_c$ in $e_l$.
            if (agent is one of the candidate answers) then
                $answer = agent$
            end if
        else if ($e_p$ in current $e_l$ contains target pronoun participation as $-o$) then
            check patient of corresponding $e_c$ in $e_l$.
            if (patient is one of the candidate answers) then
                $answer = patient$
            end if
        end if
        if (one of the event in $e_l$ is negative i.e. contains word 'not' before it in the sentence) then
            Flip the answer.
        end if
        $confidenceArr[answer] = confidenceArr[answer] + 1$
    end for
    $confidenceArr[0] = 100 * confidenceArr[0]/(confidenceArr[0] + confidenceArr[1])$
    $confidenceArr[1] = 100 * confidenceArr[1]/(confidenceArr[0] + confidenceArr[1])$
    return $confidenceArr$
end function
```

The same algorithm is run second time using the event-subject-object maps prepared from Stanford parser. It returns another confidence array for the candidate answers. We take the average of the two arrays to form final confidences on the candidate answers. The example of above algorithm is :

Example 32 *The firemen arrived after the police because they were coming from so far away .*

```
Candidate answers : firemen, police
target pronoun : they


Event    Subject    object     map :
coming   they       null
arrived  firemen    police


list of pronoun events : come
list of candidate events : arrive
```

So we call 'getConfidenceAssertionMethodConceptNet' with the above lists.

```
Returned result count : 1
(because there is an assertion come, HasProperty, arrive)
```

Hence from the returned count, we can infer that, the person who comes can also be said to be arrived. In the formal terminology, subject of come is equivalent to the subject of arrive.

```
come-s  =  arrive-s


Hence target pronoun 'they' resolves to 'firemen'.
```

The motivation for the above algorithm is described below.

1. The algorithm assumes that the two events which are strongly connected on the conceptNet i.e. the two events which have several different relationships between them will involve similar participation from the protagonist. This assumption is based on the fact that there are many relation types in the ConceptNet e.g. MotivatedByGoal, HasFirstSubevent, HasSubevent, HasContext, DerivedFrom, UsedFor etc, that exist between the two concepts and indicate that the actor of one concept must be the actor for the second event. For example :

Example 33 *Eat, MotivatedByGoal, Hungry*

This indicates that the actor who is hungry will eat, in other words the protagonist which will participate in the event Hungry as agent/subject will also participate in the event Eat as agent/subject.

2. We also consider the negative events in the above algorithm. If some event is negative i.e. it contains 'not' in the Winograd sentence before it, and it produces the answer, then we simply flip the answer obtained from it, i.e. if the answer given by that pair is agent/subject then we make it patient/object and vice versa. This is because often the presence of negation in the event will change the answer. The justification for this assumption is as follows. A Winograd sentence actually consists of two sentences connected by a discourse connective, in the first sentence the subject performs some action on object and the reasoning behind it is given in the second sentence which will be centred around the subject. In the case of 'not' associated with the action that the subject performs in the first sentence, we assume that the reasoning in the second sentence would be centred around the object because of which the subject was not able to perform the action. For example :

   Example 34 *The trophy does not fit in the suitcase because it is too small.*

   The two sentences in this example are :

   1. The trophy does not fit in the suitcase

   2. it is too small.

   and they are connected via a discourse connective because.

   To explain the example above, for justification of our approach to flipping, if there was no 'not' in the first sentence then the second sentence would be about the 'trophy' but since there is a 'not' associated with the action 'fit' we expect the second sentence would be about 'suitcase'. In other words, if there was no 'not' associated with 'fit' event then the target pronoun 'it' would have resolved to trophy, but since there is 'not' in the sentence the target pronoun resolves to 'suitcase'.

3. In the above mentioned algorithm, while calling $getAssertionCount$ function to obtain assertion counts from ConceptNet web API, we pass base verb forms of the

events(e_p, e_c). The base form is a present tense form of the word. The base forms are obtained using the JAWS utility which uses WordNet for finding a base form. The purpose of using base form here is that the event may be contained in the sentence in some other form like past participle, present continuous tense etc. and that form won't necessarily exist in the ConceptNet knowledgebase. Also converting to base form and then searching in the knowledgebase might fetch some additional useful knowledge, about the event. As shown in the 'firemen' and 'police' example above, if we search ConceptNet web API for a relation between 'coming' and 'arrived' then we would get no count. Hence we convert them to their base forms 'come' and 'arrive' and then send the query to web API.

- *Path length using Winograd sentence* : This mechanism is based on the path lengths obtained from the $getMinPath$ method described in 'Use of knowledgebase' section. We run this mechanism twice, once each for each parser's event map obtained from the sentence parsing stage. In this mechanism, we try to establish a candidate's affinity or the association with the event by finding the shortest path from each of the candidates to each of the events in the sentence. Now the peculiarity of the Winograd schema sentences, as described in chapter 1, is that the schema makers try to use proper names for the noun candidates in the sentence. This proves as an obstacle to us in this path length mechanism. Hence we first use Framenet semantic role labelling mechanism, described in section 3.5, on the events for which the candidate answer is a person name. Thus we obtain the FrameNet roles for such candidates, if the role was found for that particular event in FrameNet, and then use these roles to find affinity with the events in the sentence. We have developed a method $getConfidencePathLengthMethod$ which takes as input the FrameNet semantic roles of the candidates, list of all the events in the sentence and the list of events in which pronoun is involved and how($-s$ or $-o$). The algorithm for the implementation of this function is as follows :

Algorithm 6 $Find\ correct\ candidate\ answer\ by\ Path\ Length\ from\ sentence\ method$

```
function GETCONFIDENCEPATHLENGTHMETHOD(String $pronounEvents$, String $sentenceEvents$)
    Get FrameNet roles for the candidate answers containing proper name. Let the roles be $candidateRole1$ and $candidateRole2$.
    for ($e_s \in sentenceEvents$) do
        Call $getMinPath$ with $premise = e_s$, $alt1 = candidateRole1$, $alt2 = candidateRole2$
        store min path length for each candidate for $e_s$ in the map $hmapPathsRelations$
    end forSearch for minimum paths from candidate roles to the pronoun events.
    for ($e_p \in pronounEvents$) do
        Find the path_length(affinity) from C1 and C2 to current event $e_p$ using map $hmapPathsRelations$ constructed earlier.
        if ( path_length(C1,$e_p$) < path_length(C2,$e_p$) ) then
            if (the event is positive) then
                $answer = C1$
            else
                $answer = C2$
            end if
        else if ( path_length(C1,$e_p$) > path_length(C2,$e_p$) ) then
            if (the event is positive) then
                $answer = C2$
            else
                $answer = C1$
            end if
        else if ( path_length(C1,$e_p$) == path_length(C2,$e_p$) ) then
            Check if one of the two candidates have a lower path length event, other than $e_p$ in the map $hmapPathsRelations$.
            if (C1 has other lower path length event and C2 doesn't) then
                $answer = C2$
            else if (C2 has other lower path length event and C1 doesn't) then
                $answer = C1$
            else if (Both C1 and C2 have more affinity i.e. lower path length with other events) then
                Check if either of C1 or C2 has lower affinity to all other events than the other.
                if (C1 has more affinity to all other events except $e_p$) then $answer = C2$
                else if (C2 has more affinity to all other events except $e_p$) then $answer = C1$
                end if
            end if
        end if
        $confidenceArr[answer] = confidenceArr[answer] + 1$
    end for
    $confidenceArr[0] = 100 * confidenceArr[0]/(confidenceArr[0] + confidenceArr[1])$
    $confidenceArr[1] = 100 * confidenceArr[1]/(confidenceArr[0] + confidenceArr[1])$
    return $confidenceArr$
end function
```

The example which shows this algorithm in action is :

Example 35 *The city councilmen refused the demonstrators a permit because they advocated violence .*

```
candidate answers : councilmen, demonstrators
target pronoun : they

Event        Subject        Object     Map :
refuse       councilman     demonstrator
advocate     they           violence

Pronoun events : advocate-s
sentence events : refuse, advocate
```

Using the above inputs we invoke 'getConfidencePathLengthMethod' method. The output is as follows :

```
candidate         event      path_length
councilmen        refuse     4
demonstrators     refuse     2
councilmen        advocate   4
demonstrators     advocate   3
```

From above output we can see that the pronoun event is 'advocate' and the candidate 'demonstrators' has a closer association to it i.e. has a lower path length.
Hence since the demonstrators have closer association to pronoun event, the target pronoun is resolved to 'demonstrators'.

The above algorithm accounts for the negativity of the event by flipping the answer, in the same way as the previous one. This algorithm also uses base form of the events, for the purpose of searching in the ConceptNet, the reasoning for doing so is the same as previous inference mechanism. The algorithm returns the confidence percentage over the candidates. At the end of execution of this algorithm we'll have two such confidence arrays, one from Boxer's output and the other from Stanford parser's output. We take average over them to decide the final confidence percentage over the answers.

This algorithm basically works by finding the closeness or association, in terms of path lengths, between the candidate answer's role in the sentence and the events in the sentence. Then it chooses the closest candidate role with each event, if that event is also an event in which the target pronoun participates then the algorithm infers that the protagonist in the pronoun event i.e. the pronoun resolves to the closest candidate role, and hence we increase the confidence on that candidate as an answer. The motivation for this assumption is that the shortest path between the two concepts in the ConceptNet, will indicate the smaller justification or simpler reasoning for the connection between the two concepts.

Also if one of the two concepts is a semantic role of a protagonist and the other concept is an action to be performed, then the connection between them will include a reasoning of how the protagonist performs that action. Since we eliminate the negative relations while preparing the duplets database, the existence of a path between the two concepts denote only a positive reasoning.

- *Path length using, reason part of Winograd Sentence or Winograd question* : This mechanism again uses the path length method as the previous one. The difference between the previous method and this method is that this method considers only a certain part of the Winograd sentence instead of the whole sentence. The structure of the Winograd sentence is such that it contains two sentences joined by a discourse connective. We take advantage of this fact, and split the sentence into two parts at the discourse connective, take the second part which contains the target pronoun to be resolved. Then we reduce this part further by removing stop words, pronouns, verbs which are present/past/future tenses of is,are,do,am etc. Thus we now have a sequence of words which we call as a sentence part. Then we try to find distance between semantic roles of candidate answers/their FrameNet roles and the sentence part. This mechanism uses minimal cumulative average path length method, i.e. the sentence part is broken up into words and their individual distance from each of the candidate answers/FrameNet roles are found and summed up for each candidate. Then we take average over the number of words in the sentence part. The candidate which has lowest cumulative average path length with the sentence part is decided as an answer to the question. This approach is helpful in resolving the target pronoun, when the second part of Winograd sentence contains the reasoning about the first part. This is also helpful when the question requires factual world knowledge to find an answer.

The discourse connective in the sentence is identified with the help of its POS(Part Of Speech) tag from the parsers. Both the parsers tag the discourse connective as either CC or IN. To remove the stop words we have prepared a list of common stop words which can be checked against the word in the sentence. The pronouns are POS tagged in the sentence as PRP, verbs are tagged as VB and they can be checked against a list of all the tenses of verb 'to be'. When we form premises from the broken part sentence, we convert every word to their base form before searching for path lengths in the duplets database.

One point to note here is that not every single Winograd sentence contains a discourse connective, but its a common property of the Winograd sentences. In the case when the sentence doesn't contain the discourse connective, we take question as the broken sentence part and do the same processing, as above, on it.

The algorithm for the method $getConfidencePathLengthBrokenSentence$, implementing the above functionality, is as follows :

Algorithm 7 *Find correct candidate answer by minimum cumulative Path Length from sentence or question*

```
function GETCONFIDENCEPATHLENGTHMETHOD(String $sentence$,String $question$)
    if ($sentence$ contains discourse connective tagged as $CC$ or $IN$) then $sentencePart$ = substring of $sentence$ from discourse connective till the end.
    else $sentencePart$ = $question$
    end if
    for ($word \in sentencePart$) do
        if ($word \in stopWordList$ OR $word$.getPOSTag() is $PRP$ or $TO$ or $WP$ or $UH$) then
            remove $word$ from $sentencePart$.
        else if ($word$.getPOSTag() has substring $VB$) then
            get base form of $word$.
            if ($baseForm \in ["is","are","am"]$) then
                remove $word$ from $sentencePart$.
            end if
        end if
    end for
    for ($word \in sentencePart$) do
        add $word$ to list premise $p_l$.
    end for
    Get FrameNet roles for the candidate answers containing proper name. Let the resolved roles be $candidateRole1$ and $candidateRole2$.
    Call $getMinPath$ with $premise = p_l$, $alt1 = candidateRole1$, $alt2 = candidateRole2$
    for ($p \in P_l$) do
        if ($min_path(candidateRole1, p) <= 4$) then
            if ($isNegativeEvent(p)$) then
                $min_path(candidateRole1, p) = 0$
            end if
            $pathLength[candidateRole1] = pathLength[candidateRole1] + min_path(candidateRole1, p)$
        end if
        if ($min_path(candidateRole2, p) <= 4$) then
            if ($isNegativeEvent(p)$) then
                $min_path(candidateRole2, p) = 0$
            end if
            $pathLength[candidateRole2] = pathLength[candidateRole2] + min_path(candidateRole2, p)$
        end if
    end for
    if ($pathLength[candidateRole1] > pathLength[candidateRole2]$) then
        return $candidateRole2$ with $confidence = pathLength[candidateRole2]$ / ( $pathLength[candidateRole1]$ + $pathLength[candidateRole2]$ )
    else if ($pathLength[candidateRole1] < pathLength[candidateRole2]$) then
        return $candidateRole1$ with $confidence$ = $pathLength[candidateRole1]$ / ( $pathLength[candidateRole1]$ + $pathLength[candidateRole2]$ )
    end if
end function
```

The example of the above algorithm in action is as follows :

Example 36 *The councilmen refused the demonstrators a permit because they advocated violence .*

```
Pronoun to be resolved : they
Candidate Answers : Councilmen, Demonstrators

Second sentence part : because they advocated violence
Broken sentence part : advocate violence

Input to program :
%premise
premise_contains(advocate).
premise_contains(violence).
%alternative 1
alternative1_contains(councilman).
%alternative 2
alternative2_contains(demonstrator).

Output :

Answer: 1
alternative1_relation(councilman,violence,5)
alternative1_relation(councilman,violence,4)
alternative1_relation(councilman,advocate,5)
alternative1_relation(councilman,advocate,4)
alternative2_relation(demonstrator,violence,5)
alternative2_relation(demonstrator,violence,4)
alternative2_relation(demonstrator,violence,3)
alternative2_relation(demonstrator,advocate,5)
```

```
alternative2_relation(demonstrator,advocate,4)

alternative2_relation(demonstrator,advocate,3)

SATISFIABLE




Cumulative path lengths :

            councilman      demonstrator

advocate        4               3

violence        4               3


Cumulative      8               6


number of words in broken part : 2

hence --

average         4               3


Answer :

min cumulative average path : 3  for the candidate 'demonstrators'


Hence target pronoun resolves to -> Demonstrators
```

This method like the previous inference mechanisms, returns an answer and confidence over it, it also takes into account the negative events, i.e. if there is a word 'not' in the sentence before the event and $getMinPath$ finds a path from a candidate to the positive event, then the path length is considered as 0 for the negative event.

As opposed to the other inference mechanisms described above, this mechanism only runs once because it doesn't consider event maps or anything else which is parser specific. The POS tags are considered from the Stanford parser's output.

Note that in all three of the inference mechanisms mentioned above, if we don’t find any path or get 0 assertion count for one of the candidate answers but the other candidate answer has paths or the positive assertion count, then we don’t proceed with the algorithm, instead leave the resolution unanswered for that Winograd sentence for that inference mechanism.

Finally since we get three different answers from three ConceptNet inference mechanisms described above, we need a combining mechanism to converge them into one answer. We just use an averaging method for this purpose. We obtain the final confidence arrays from all the three mechanisms ,average out the confidences over both the candidates and declare the final answer based on, which candidate has more average confidence.

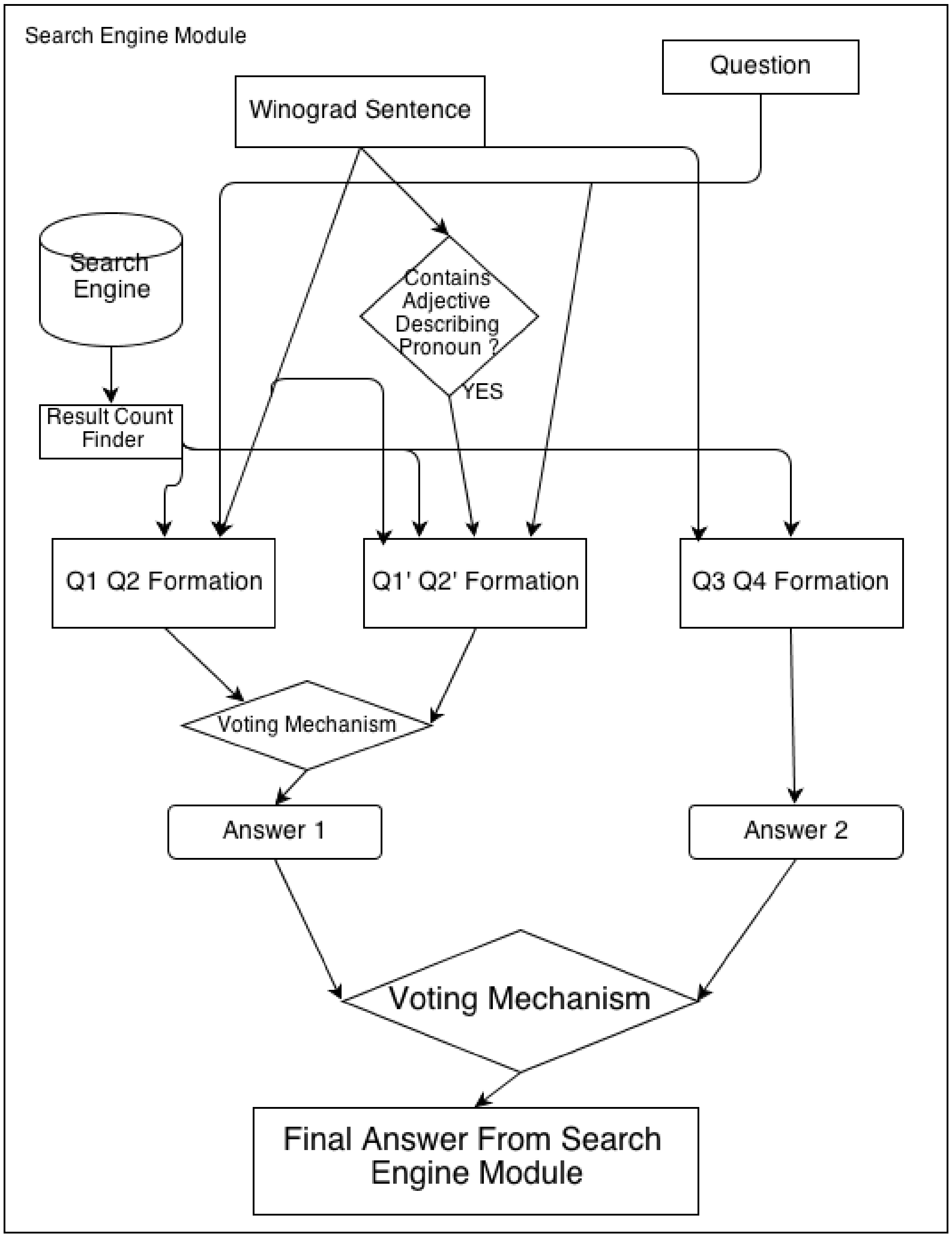


Figure 3.5: Overview of the Search Engine Component

Why Search Engines ?

The search engines are probably the largest available text corpus of real world knowledge, today. There is a large amount of real world knowledge that can be harvested through search engines and used in resolving target pronoun in a Winograd schema sentence. The property of Winograd sentences, as described in section 1.2, is that with change of only a single word, the answer changes. Hence the challenge is devise some intelligent techniques to use this huge text corpus to draw inferences in some way, because traditional statistical techniques over the large text corpus won't work correctly in this case.

Another Winograd schema property is that it is not domain specific and covers a broad range of human reasoning. This demands the dynamic knowledgebase which provides updated information on various domains. Such an information is provided by the search engines because the web pages they search across contain large amount of text information on different topics. The search engines can be queried using the 'lexico-syntactic query' and they return the page results along with the estimated count of web pages containing part or whole of that search query. Through these counts we can infer that certain query pattern is more frequent than certain other query patterns that return lower counts. Thus through semantic search queries we can learn the pattern of word usage or the pattern of word ordering, in the sentence, which in turn can be used to make inferences. If we can form such a chain of inferences on the Winograd sentences, then that will lead us to resolving the target pronoun to the correct candidate answer.

Also there are some obvious connections that can be made from the search engine results. For example

Example 37 *The knife sliced through the flesh because it was sharp.*
In this case humans resolve the pronoun 'it' to 'knife' by using the background knowledge that the property 'sharp' is associated more to the 'knife' than 'flesh'. Also if we try to search in the search engine we get following result counts :

```
Query : knife sharp
count of results : 41,100,000
```

```
Query : flesh sharp
count of results : 20,500,000
```

As can be seen, the count for the first query is greater than the count for the second query by more than 100%. Thus from these result counts we can draw inferences about the associations of terms.

To automate the process by which humans resolve the situation in the sentence using the background knowledge, we have to develop an algorithm which can form meaningful search queries to search in the search engines.

Use of Search Engines in our research

We use two search engines, Google and Bing, in our research to obtain the result counts. This is to find an answer if one search engine is unable to choose between the two search patterns. To form meaningful search queries, we use the structure of the Winograd sentences. It is such that, the Winograd sentences contain two parts connected via a discourse connective. The first part contains the two candidate answers and the second part contains the pronoun to be resolved. Also another fact is that many a times the Winograd question almost resembles the second part of the sentence. We use these properties for search queries formation and inference of results as follows :

- *Formation of Q1 and Q2* : Find a discourse connective D in the sentence such that, both the candidate answers are in the sentence part before the connective D and D is tagged as either 'IN' or 'CC' in Stanford parser's POS tag output. For example, in the sentence – The man could not lift his son because he was so weak . – The discourse connective D would be the word 'because'. We can break the sentence on this discourse connective D, and consider second part – he was so weak – for the query formation.
  If such a discourse connective is not found in the sentence then we search for the target pronoun in the sentence and see if we can break the sentence in two parts at that index. For example in the sentence – The drain is clogged with hair , it has to be cleaned . – There

is no discourse connective satisfying the conditions. Hence we consider pronoun 'it' as the breaking point for the sentence. The second part – it has to be cleaned – is considered for query formation.
If above two methods fail to generate a part of the sentence that is atleast two words long, then we use the Winograd question to form the queries. We consider the 'wh' part of the question equal to the pronoun to be resolved.
From the sentence or question part gathered above, we form the $Q1$ and $Q2$. For that we first form the broken form of the search query. For forming the broken part, we gather the part which we got from above, and process it word by word, rejecting all the punctuation marks, words POS tagged as 'DT'(e.g. 'a','the' etc) and words POS tagged as 'CC'(coordinating conjunctions e.g. : but, because etc.). Also when processing we encounter the target pronoun(or the 'wh' part from the question), we replace it with characters '##'.
After the broken part is formed we form $Q1$ and $Q2$ by replacing the '##' part in the broken part with the candidate answers, $candidate1$ and $candidate2$, respectively. For example :

Example 38 *I put the cake away in the refrigerator , it has a lot of leftovers inside .*

```
The above sentence doesn't have the discourse connective, hence we
break the sentence at the pronoun to be resolved i.e. 'it'.

Take, second part of the sentence : it has a lot of leftovers inside

This part is POS tagged in the Stanford parser's output as :
it/PRP has/VBZ a/DT lot/NN of/IN leftovers/NNS inside/RB

Broken part formed from this part is :
## has lot of butter inside

Search queries :
Q1 : cake has lot of butter inside
Q2 : refrigerator has lot of butter inside
```

- *Formation of Q1' and Q2'* : Sometimes Winograd sentence's second part/question part, from the above, contain some adjective which describes the target pronoun. From the adjective we can determine the association patterns as described in the above section in the 'knife' and 'flesh' example. That is we want to see if the adjective is more apt to one candidate than the other candidate. In the 'knife' and 'flesh' example, the adjective 'sharp' is clearly more associated with 'knife' that the 'flesh'. But sometimes the sentence part contains such an adjective that it can't be clearly associated with any one candidate. Example of one such sentence is :

  ```
  I could not put the pot on the shelf because it was too tall .
  ```

  We form two search queries Q1' and Q2' for such sentences which contain an adjective, Adj, describing the target pronoun. The two search queries are formed as follows :
  Q1' = $candidate1$ more Adj than $candidate2$
  Q2' = $candidate1$ more Adj than $candidate2$
  This can be illustrated with an example as follows :

  Example 39 *I could not put the pot on the shelf because it was too tall .*

  ```
  The candidate answers in this sentence are
  candidate1 = pot
  candidate2 = shelf

  In the second part of the sentence i.e. "it was too tall" there is an
  adjective 'tall' describing the pronoun 'it'.
  Hence we proceed to form Q1' and Q2'.

  Search queries :
  Q1' : shelf more tall than pot
  Q2' : pot more tall than shelf
  ```

Note that we only form the queries Q1' and Q2' when the second part of the sentence contains an adjective describing more about the target pronoun. If there is no adjective, we don't form Q1' and Q2'.

- *Using Q1,Q2 and Q1',Q2' to get $answer1$* : To get the $answer1$ from the first part, we obtain opinions from Q1,Q2 and Q1',Q2' differently and then combine them to form final answer. This works as follows :

  1. We first obtain counts of results for Q1,Q2 such that
     count of results for Q1 = C1
     count of results for Q2 = C2
     Based on the above counts we decide the opinion of Q1,Q2 as follows :

     ```
     Rule 1 :
     If C1 > C2 by Z% then Ans(Q) = candidate1
     ElseIf C1 < C2 by Z% then Ans(Q) = candidate2
     Else Ans(Q) = No Decision
     ```

     The Z used in the above rule is a threshold, which makes sure that one candidate is preferred over the other only if there is a sufficient difference between the counts returned by the search engine, for their respective search queries. And since Q1 and Q2 don't share all the words, we assume that their query counts might be affected by the high occurrence of one of their uncommon terms. Hence we set threshold for Q1,Q2 to be, Z = 100. The example in action, for the procedure mentioned above :

     Example 40 *I could not put the pot on the shelf because it was too tall .*

     ```
     As we saw in the formation of Q1,Q2, the Search queries are :
     Q1 : shelf was too tall
     Q2 : pot was too tall


     The search engine counts are :
     C1 = 2440000
     C2 = 3070000
     ```

```
Now applying above rule, Rule 1,
since C2 > C1 by only 25 %
Ans(Q) = No Decision
```

2. Similar to the above procedure, we obtain counts, C1', C2' for Q1' and Q2', if the queries Q1' Q2' are formed from the adjective. The rule to decide answer for Q1',Q2' is :

```
Rule 2 :
If C1' > C2' by Z'% then Ans(Q') = candidate1
ElseIf C1' < C2' by Z'% then Ans(Q') = candidate2
Else Ans(Q') = No Decision
```

Similar to the above, Z' is the threshold for Q1', Q2'. But the value of Z' is different from Z, because Q1' and Q2' both contain the same words in them with just the positions changed. Hence greater occurrence or lesser occurrence of one of the query terms will affect both of them equally. Hence we keep the threshold Z' = 1. The example for Q1' and Q2' is as follows.

Example 41 *I could not put the pot on the shelf because it was too tall .*

```
As we saw in the formation of Q1' and Q2', the Search queries are :
Q1' : shelf more tall than pot
Q2' : pot more tall than shelf

The search engine counts are :
C1' = 4850000
C2' = 1.32E8

Now applying above rule, Rule 1,
since C2' > C1' by more than 1 %
Ans(Q') = pot
```

3. Now that we have both Ans(Q) and Ans(Q'), we combine them to find $answer1$ via following rule -

   ```
   Rule 3 :
   If Ans(Q) = Ans(Q') then answer1 = Ans(Q) = Ans(Q')
   ElseIf Ans(Q) = No Decision & Ans(Q') != No Decision
   then answer1 = Ans(Q')
   ElseIf Ans(Q') = No Decision & Ans(Q) != No Decision
   then answer1 = Ans(Q)
   Else answer1 = No Decision
   ```

   For example, in the 'shelf' and 'pot' example shown above, Ans(Q) = No Decision and Ans(Q') = pot. Hence applying the above rule, Rule 3, answer1 = 'pot'.

- *Formation of Q3 and Q4 to get* $answer2$: As stated above, we try to break the Winograd sentence into two parts based on either the discourse connective or the target pronoun. We'll call the part before the discourse connective/target pronoun as the 'initial part' and the part after, as the 'later part'. These two parts combined, and unnecessary words removed, help us in the formation of Q3 and Q4. The broken initial part formation along with the formation of Q3, Q4 is explained below :

  1. *Broken initial part formation* : The initial part is the part of the sentence before discourse connective/pronoun. This part contains the candidate answers along with the events in which they participate, and it may also contain some additional information about the candidates. For forming the broken initial part which will remove unnecessary words from the initial part, we process the sentence word by word starting from the first word till the end of initial part.
     When processing initial part word by word, we include those words from it in the broken initial part which are candidate answers or the events in which candidate answers participate. We also include the other words from the sentence which are tagged as the events by the parser, because they might not include candidate answers but they might tell us some additional information about them. We also include those words from initial

part, which are tagged as adjectives or nouns or adverbs. All the other words along with any forms of the verbs 'to be','have','do' are not included.
The example of broken initial part formation is :

Example 42 *I could not put the pot on the shelf because it was too tall .*

```
Candidate answers are : pot, shelf
pronoun to be resolved : it
discourse connective : because


Initial part : I could not put the pot on the shelf
corresponding POS tags :
I/PRP could/MD not/RB put/VB the/DT pot/NN on/IN the/DT shelf/NN


Broken initial part = B_i = I not put pot on shelf
```

At the end of this process what we have is called as the broken initial form, B_i.

2. *Formation of Q3, Q4* : For forming the queries Q3 and Q4, what we have is a broken initial form B_i, a discourse connective D, target pronoun to be resolved P and the sentence part after pronoun S_p. From these first we form temporary query :
tempQ = B_i D any adjectives, nouns, adverbs in between P S_p
From this we form Q3, Q4 by replacing target pronoun P with the candidate answers :
Q3 = B_i D any adjectives, nouns, adverbs in between $candidate1$ S_p
Q4 = B_i D any adjectives, nouns, adverbs in between $candidate2$ S_p
This can be more clearly understood with the example below :

Example 43 *I could not put the pot on the shelf because it was too tall .*

```
Candidate answers are : pot, shelf
pronoun to be resolved : it
discourse connective : because


B_i = I not put pot on shelf
```

```
D = because

P = it

S_p = was too tall


The search queries Q3, Q4 formed as described above, are :

Q3 : I not put pot on shelf because pot was too tall

Q4 : I not put pot on shelf because shelf was too tall
```

3. *Obtaining* $answer2$ *from Q3 and Q4* : To obtain $answer2$ from Q3 and Q4, we first obtain their result counts from search engine –
   C3 = count(Q3)
   C4 = count(Q4)
   Then we find the answer according to the rule described below.

   ```
   Rule 4 :

   If C3 > C4 by X% then answer2 = candidate1

   ElseIf C3 < C4 by X% then answer2 = candidate2

   Else answer2 = No Decision
   ```

   The role of X here is the same as previous rules, that of the threshold. Since the search queries Q3 and Q4 tend to be big in size many times, we keep X = 1. Also all the search terms are common in both search queries hence there is no issue of one term increasing result of one query.
   Continuing with the 'pot' and 'shelf' example in this section, the $answer2$ from Q3, Q4 for that example is :

   Example 44 *I could not put the pot on the shelf because it was too tall .*

   ```
   Candidate answers are : pot, shelf

   pronoun to be resolved : it


   Q3 : I not put pot on shelf because pot was too tall
   ```

```
Q4 : I not put pot on shelf because shelf was too tall


C3 = 2.17E7

C4 = 1.21E7


According to rule 4, since C3 > C4 by more than 1%,

answer2 = pot
```

- *Combining $answer1$ and $answer2$ to get final answer $answer$* : To get the final answer - $answer$, from the search engine module we combine the two answers obtained from the sub modules as follows :

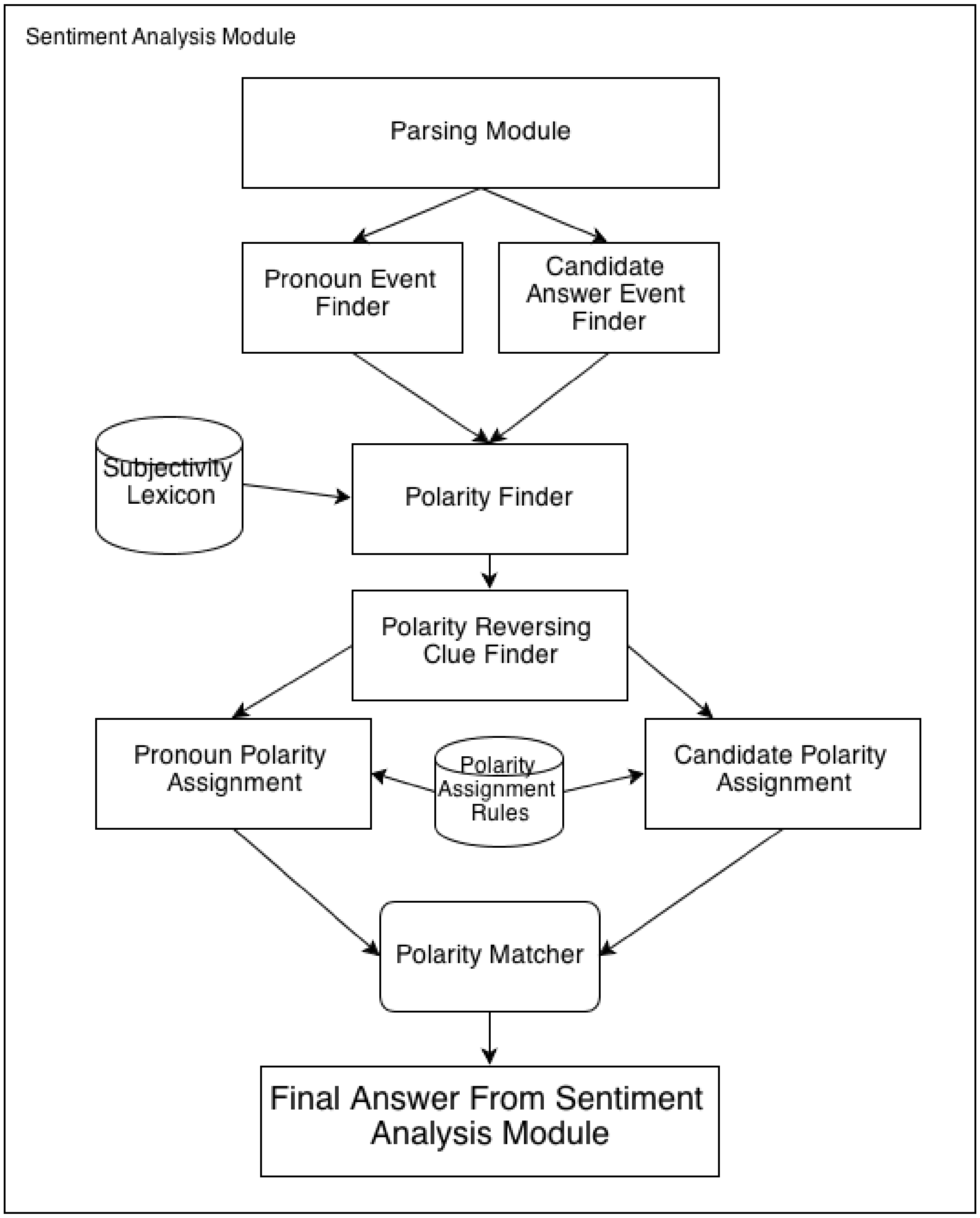
  ```
  Rule 5 :
  If answer1 = answer2 then answer = answer1 = answer2
  ElseIf answer1 = No Decision & answer2 != No Decision then answer = answer2
  ElseIf answer2 = No Decision & answer1 != No Decision then answer = answer1
  Else answer = No Decision
  ```

  Continuing with the 'pot' and 'shelf' example above, since $answer1$ and $answer2$ are both 'pot', the final answer would be $answer$ = pot.

Note that we always use Stanford parser's POS(Part Of Speech) tags for the formation of queries Q1,Q2,Q1',Q2' and we use two different parsers, Boxer and Stanford parser, for the formation of Q3,Q4 and then combine their results.
Also when the candidate answers are person names, we use their semantic roles obtained from the FrameNet module to form search queries.
Other researchers have also used lexico-syntactic search query patterns for anaphora resolution, e.g. poesio et al. 2004 or Modjeska et al. 2003. But these researchers use them for targeting specific relations e.g. P of Q (wheel of car) etc. But our search query patterns are sentence specific and general enough to capture much of the information in the sentence. The patterns like

Q3, Q4 or Q1, Q2 in our research don’t try to target any specific relation since it is not possible to always fit the required world knowledge in particular relations.

3.9 Use of Sentiment Analysis



Sentiment Analysis Module
Parsing Module
Pronoun Event Finder
Candidate Answer Event Finder
Subjectivity Lexicon
Polarity Finder
Polarity Reversing Clue Finder
Pronoun Polarity Assignment
Polarity Assignment Rules
Candidate Polarity Assignment
Polarity Matcher
Final Answer From Sentiment Analysis Module

Figure 3.6: Overview of the Sentiment Analysis Component

Why Sentiment Analysis ?

Sentiment analysis is the task of identifying positive, negative or neutral opinions or emotions. There are various types of sentiment analysis : document level, sentence level, phrase level. We use phrase-level sentiment analysis in our research here. Let's see how humans use phrase level sentiment analysis in resolving the target pronoun in Winograd sentences.

Example 45 *The city councilmen refused the demonstrators a permit because they feared violence .*
Let's see how humans resolve the pronoun 'he' to 'Councilmen'. Here the word 'refuse' has a negative sentiment. Hence the one who is being refused is the negative character i.e. he has the negative polarity, in other words the object corresponding to the event 'refuse' will have the negative polarity. The person who does the refuse action is the positive character in this story i.e. he has the positive polarity. That means the subject of event 'refuse' has the positive polarity.
Similarly, fear has the negative sentiment. That indicates that some good guy fears bad guy, i.e. in other words some positive character fears the negative character. That also means that the subject of 'event' fear has a positive polarity and the object of fear has the negative polarity. In this sentence subject of 'fear' is 'they'. Hence the word 'they' has the positive polarity.
Since 'they' are positive people in this sentence, word 'they' is resolved to the 'councilmen' who are also positive people in this sentence.

We want to automate this human thinking process into an algorithm which we can use to correctly find an answer to the Winograd sentences. We have Wilson et al.'s (2005) subjectivity lexicon which contains 8221 words labeled with the polarity(positive, negative or neutral) and their use in the sentence(Part Of Speech tag). This lexicon can be used in automating the above algorithm.

Use of Sentiment Analysis in our research

We have seen in the above section, how the humans use thinking in resolving the target pronoun in the Winograd sentences. We automate this process, using the above mentioned subjectivity lexicon, as follows :

- From the event map obtained in the sentence parsing phase, detect the event/events in which candidate answers participate.
- Find polarity of these events from the subjectivity lexicon. From the event's polarity and the candidate's role in that event(subject or object), assign polarities to the candidates.
- Find the event in which the target pronoun participates. Find the polarity of the event. Assign polarity to the target pronoun, according to its role in that event(subject or object).
- Resolve the target pronoun to the candidate answer which has the same polarity. If there is no candidate with same polarity then the answer is 'No Decision'.

The working example of the above mentioned pseudo-code is as follows :

Example 46 *The city councilmen refused the demonstrators a permit because they advocated violence .*

```
In this example 'they' is the target pronoun. And 'councilmen' and
the 'demonstrators' are the candidate answers.

The event maps from Parsing module are as follows :

Event       Subject     Object      Map :
advocated   they        violence
refused     councilmen  demonstrators



This means following :
councilmen = subject of the event refuse
demonstrators = object of the event refuse
they = subject of the event advocate
```

The pronoun 'they' is resolved to correct candidate answer using the steps in the pseudo-code mentioned above as follows :

- *Both the candidate answers participate in the event 'refuse'.*
- *We find the polarity of refuse event. The polarity given in subjectivity lexicon is NEGATIVE. Hence the polarity of the object i.e. 'demonstrators' becomes negative and polarity of the subject 'councilmen' becomes positive.*
- *The target pronoun 'they' participates in the event 'advocate'. The polarity of the event 'advocate' in the subjectivity lexicon is POSITIVE. Since target pronoun is the subject of the event 'advocate', its polarity becomes negative.*
- *As we see from the above that the polarity of both the target pronoun and the candidate 'demonstrators' is the same. Hence we resolve target pronoun 'they' to the candidate 'demonstrators'.*

Some details about the above mentioned algorithm are as follows :

- *The polarity reversing for candidate event depending upon the context* : The candidate event/events are the ones in which the answer candidates participate. Suppose there is a candidate event Ea with polarity Pa. Now we look for clues in the given sentence that might flip the polarity of this event. The flip happens if the event Ea is negated with the word 'not'. Hence before assigning polarity to any candidate event, we check whether the polarity is be flipped by the usage of word 'no' or 'not' before the event in the given sentence. If there is word 'no' or 'not' in before the event then the polarity of the current event Ea becomes 0-Pa. Note that if Pa is detected to be a neutral event, then its polarity is not flipped but instead made negative.
- *The polarity reversing for pronoun event depending upon the context* : Similar to the process of flipping mentioned above, we also search for polarity flipping clues in the sentence, for the pronoun event. In addition to the words 'no' or 'not' before, which flip the polarity, we search for one more clue to flip the pronoun event polarity – negative discourse connective.

As we know, many of the winograd sentences contain the discourse connectives i.e. the conjunctions. Some of them are negative e.g. : though, even though, against, but etc. And these change the pronoun event's polarity, if they come before the pronoun event in the sentence. For example : The woman held the girl against her will . The event in the above sentence, 'will' has the positive polarity in the subjectivity lexicon, but it is negated in the sentence by the use of discourse connective 'against', hence overall polarity of the event 'will' becomes negative. Similarly the discourse connective 'but' indicates that the part of the sentence that is coming up contains disagreement with the part before. Also while doing this flipping, if the pronoun event has neutral polarity and the flipping word is detected then the polarity of the event is made negative.

- *The polarity assignment* : The main tricky part in this algorithm is assigning polarities to the answer candidates and target pronoun.
  For the candidates, we consider the events that they participate in, from the event maps generated by the parsing module. The candidate events are of the type $verb$. From the examples seen above, it can be seen how the verbs are treated for polarity assignment. Since the verb is an action that is applied onto the object, we assign object of the verb with the same polarity as the verb event. The polarity of the verb can be thought of as the state of the object after the action is applied or taken place on it. Oppositely the action is applied by the subject of that verb, hence subject of the verb is assigned the opposite polarity. As shown in the example in the previous section, $refuse$ has the negative polarity and hence its object 'demonstrators' gets the negative polarity and the subject 'councilmen' gets the positive polarity.
  For the target pronoun, the events from the event maps can be of the two types mainly, $verb$ or $adjective$. They are treated differently for polarity assignment. The events of the type verb are treated in the same way as described above for the candidate events. That is, if the target pronoun participates in the verb event as the object then it would get the same polarity as the verb and opposite polarity if it participates as the subject. Opposite is the case of the events tagged as 'adjective' by the parser. In case of the adjectives, it provides more description about the subject participant. For example : The lawyer asked the witness a question , but

he was reluctant to repeat it . In this sentence the target pronoun 'he' participates in the event 'reluctant' as the subject. The word 'reluctant' is tagged as the adjective. It describes some property of the pronoun 'he'. Here 'reluctant' has the negative polarity and hence the target pronoun 'he', which participates in the adjective event as the subject, is also assigned the negative polarity.

Sometimes due to parser's inability to parse some complex sentences, the target pronoun event is not detected from the sentence. In that case we try to polarize the target pronoun using the words from the question. For example :

Example 47 *The sculpture rolled off the shelf because it was not level .*

```
The boxer parses this and prepares event map as follows :


Event   Agent       Patient Map :
roll    sculpture   shelf


polarity of the event 'roll' = NEGATIVE
hence from the above described logic :
        polarity of the candidate answer 'sculpture' = POSITIVE
        polarity of the candidate answer 'shelf' = NEGATIVE


The event map doesn't contain any pronoun event.
In such a case we consider the question.
```

question : What was not level ?

```
In this case level is tagged as an adjective by the parser.
And it describes something about 'what', which corresponds to the pronoun,
'it', in the sentence. Hence we use the polarity of the adjective 'level'
in the question, to decide the polarity of the pronoun 'it'.
```

```
The polarity of 'level' = NEUTRAL
But it contains word 'not' before hence its polarity becomes negative.
As it describes something about the target pronoun 'it',
we can assign the same polarity to the target pronoun.
Hence the polarity of the target pronoun becomes NEGATIVE.


Hence from the above described logic of answer finding, the pronoun
resolves to 'shelf' because both have NEGATIVE polarity.
```

Also note that when the polarity of the verb is found to be NEUTRAL and there is nothing which changes the polarity, we assign the neutral polarity to both the candidates.
In the similar cases as above where no pronoun event is detected, we use words tagged as adjectives or nouns or adverbs from the question. These words describe more about the 'wh' part of the question, which corresponds to the target pronoun in the sentence.

The above procedure is conducted two times, once for each parser, Boxer and Stanford parser, because many a times these two parsers detect different candidate events and pronoun events due to their different parsing styles. Hence we take care that the sentiment analysis module is run for both parsers and then we obtain two answers from Boxer and Stanford parser as A_b and A_s respectively. We combine these two answers to find the final answer,$AnsSentiment$ from sentiment analysis module by applying following rule :

```
Rule :
If A_b = A_s then AnsSentiment = A_b = A_s
ElseIf A_b = No Decision & A_s != No Decision then AnsSentiment = A_s
ElseIf A_s = No Decision & A_b != No Decision then AnsSentiment = A_b
Else AnsSentiment = No Decision
```

3.10 Result Classification and Obtaining final answer

As shown in the system architecture diagram 3.1, our system has 4 answer finding modules, namely ConceptNet, Narrative chains schema, Search engines, Sentiment analysis. We obtain four different answers : $AnsCnc$, $AnsNc$, $AnsSe$, $AnsSan$ from them respectively. The classification algorithm combines these answers to finalize one answer $finalAnswer$ for the system. For combining the answers we assign equal weight to each module's answer. Then we calculate the number of modules $N1$ which have $candidate1$ as the answer, and number of modules $N2$ which have $candidate2$ as the answer. Then if $N1 > N2$ then we declare $finalAnswer = candidate1$. Else if $N1 < N2$ then we declare $finalAnswer = candidate2$. Also if $N1 = N2$, then we have following rule to find out the final answer.

```
Rule :
If N1 = N2 then
    if AnsSe != No Decision then
        finalAnswer = AnsSe
    else if AnsNc != No Decision
        finalAnswer = AnsNc
    else if AnsSan != No Decision
        finalAnswer = AnsSan
    else if AnsCnc != No Decision
        finalAnswer = AnsCnc
EndIf
If no answer yet then
    finalAnswer = Answer from Q3,Q4 of Search Engine Module
```

The above rule states that if $N1 = N2$, then we prioritise the modules to decide the final answer. In this procedure, the modules according to descending priority are Search engine, Narrative chains schema, Sentiment analysis, ConceptNet.

Chapter 4

Evaluation of system on Original Winograd Corpus

4.1 Experiment Setup

Experiment Dataset

To evaluate our system, we use the original winograd schema corpus (by Levesque, 2011). It is available for download in the standard text format. It contains 260 entries, where each entry consists of a Winograd sentence and a question. Out of these 260 sentences, our classifier rejects 170 entries because the sentence in the entry contains a person name which requires semantic role resolution. Hence our dataset actually consists of only 90 entries from the original dataset.

Configurations

We have thresholds at some places. For every place in which the word similarity or sentence similarity mechanism, described in section 3.2, is used we keep threshold as 50%. That means we consider the word or a sentence for evaluation only if it is equal to or more than 50% similar to the current input word or sentence. For example, when we evaluate similarity between the events in Narrative chains module the rule is that the event is considered for evaluation when its similarity to current event is 0.5 or more. Also we form three types of search queries in our Search engine module. We use threshold of 100% for the queries Q1,Q2. And we use threshold of 1% for the queries Q1',Q2' and Q3,Q4.

Evaluation Criteria

The main aim of the system is to correctly answer as many Winograd questions as possible, or in other words correctly resolve as many target pronouns in Winograd sentences as possible. We evaluate the system's efficiency based on the percentage of correct answers or correctly resolved target pronouns. We assume that the system is always forced to resolve the target pronoun, and even after that if some answer stays as a "No decision" then we split it as half correct answer and half wrong answer. We measure the percentage of correct answers and wrong answers.

Table 4.1: Comparison of results from UTD's pronoun resolver, random answer system and our system

System	Correct Answers	Wrong Answers	No decisions
UTD Pronoun Resolver	68.88%	31.12%	0.0%
Random Answer System	50%	50%	0.0%
Our system	73.33%	26.67%	0.0%

4.2 Comparison with other systems

- *Randomiser or Random Answer System* - This is the first baseline system that we compare against. This system will randomly guess the answer. Since there are two possible answers to each of the Winograd sentence, the possibility of this system getting the correct answer is 50%, as is the probability of wrong answer.

- *UTD's pronoun resolver System* - Our second baseline system is University of Texas at Dallas's pronoun resolver system for definite pronouns($Rahman, Ng\ 2012$). This system, like us, uses knowledge-rich approach to resolve the target pronouns. Their system has 8 features which use inference and reasoning to find the answer individually and then combines the answers using ranking based machine learning technique. Their system achieves 67.9% accuracy on the whole Winograd schema. But to ensure fair comparison with our system, we test its results on the same classified dataset that we run our system on. In the confined domain their system resolves 68.88% pronouns correctly and 31.12% pronouns incorrectly.

4.3 Results and analysis of system performance

We present the system's performance results on the classified Winograd dataset of 90 entries. Here we compare our results with University of Texas at Dallas's pronoun resolver system ($Rahman, Ng\ 2012$) and the random answering system.

As can be seen from the above table, our system performs better than the randomiser system by 23.3 % and it also outperforms UTD's pronoun resolver system by 4.45 % in this confined version of the Winograd schema.

Chapter 5

Conclusion, Improvements and Future Work

The research related to anaphora resolution or co-reference resolution has been going on for many years. The co-reference resolution problem has gained recent interest due to the Winograd schema challenge and the fact that the Winograd schema challenge is presented as an alternative to the Turing test. It poses a challenge that requires the system to draw large amount of inferences, something which static, rule based systems such as the Stanford Resolver(Lee et al., 2011) are not able to do. We need new knowledge sources for this challenge. The current state of the art for co-reference resolution is the UTD's pronoun resolver system. We have tried to go a step further with the use of new knowledgebase ConceptNet and some new inferencing techniques, and we have got better results in a confined domain. There are lot of improvements that can be done to this system. Some of them are noted below.

- One improvement that can be made is regarding query formation in the search engine module. The search engines are great at representing factual knowledge, but they don't work well for handling associations between the things. The presence of words in the search query one after the other doesn't always guarantee an existence of any kind of relationship between them. For example in the Winograd sentences we are looking for the subject-verb-object relationships, these in particular aren't captured by the search engines. Our search queries try to avoid that problem by capturing as much reasoning from the sentence as possible, but that results in a very low number of hits due to sparse data. The improvement can be made to carefully construct the search pattern such that we don't miss out on important information from the sentence but the query is able to get hits.
  Also other improvement in this regard is about the formation of query from the sentence. Currently we form the search query with the help of POS tags from the parser, this method doesn't always form the optimal search query. Investigating more into the sentence structures and then including particular POS tags from a sentence might be a way to go.
- The Winograd sentence is seen as a sequence of events in the sentence. Narrative chains

schema helps us in making inferences by using this chain of events. However Narrative chains schema is just computed over verbs and hence misses out on a large number of pronoun resolutions. For example

Example 48 *The table would not fit through the doorway because it is too wide .*

In the above sentence, there is an event 'fit' and it is affected by an adjective 'wide' associated with the pronoun. If we can make some sort of connection between 'fit' and 'wide' then resolving target pronoun is easy. One improvement that can be made in this regard is capturing how adjectives that describe a pronoun in the sentence affect the events in which candidate answers are involved.

- Another improvement can be made with regards to the use of ConceptNet knowledgebase that we are using. ConceptNet has various relations between concepts expressed as a hypergraph. But a particuar relationship that we are searching for, might not always exist. For example, when we search for a relationship between fit and small, we don't get any assertion. But there exists an assertion : "fit,HasProperty,perfect_size". If we can use this somehow to draw the inferences, then ConceptNet will be a very useful resource in solving Winograd like problems which require use of a background knowledge.

- Both the parsers that we use, the Stanford parser and Boxer parser, are shallow parsers. For improving the efficiency, deep parsing can be done. For example consider two sentences below :

Example 49 *1. Jim defeated John because he was less popular.*
2. Jim was defeated by John because he was less popular.

In both the above sentences, 'defeated' is the event in consideration but its subject and object are reversed. Currently Stanford parser and Boxer, both don't correctly detect that. That hampers the efficiency of our system in Narrative chains module and Sentiment analysis module, because detecting syntactic roles of candidate answers is so important to both these modules.

Also on the similar lines, the parsers currently can't parse the subjects or objects which consist of multiple words. An example of this is:

Example 50 *Everyone really loved the oatmeal cookies; only a few people liked the chocolate chip cookies and next time, we should make more of them .*

In this example, oatmeal cookies and chocolate chip cookies are the candidate answers. But since parsers don't detect the whole term as a subject/object to any event, we had to modify the sentence as :

Example 51 *Everyone really loved the oatmeal_cookies , only a few people liked the chocolate_cookies , Next time , we should make more of them .*

We would like to improve our system on the above lines and in future we would like to convert our system into a general purpose pronoun resolver and evaluate it on large real world corpora like MUC.

Chapter 6

Appendix

6.1 Detailed Comparison between UTD pronoun resolver and our system

Here is a module by module comparison between our system and the UTD pronoun resolver system.

Table 6.1: Comparison of Answering modules from UTD's pronoun resolver and our system

No.	Our System	UTD's System
A.	Common Modules	
1.	Narrative Chains Module	
	Our system uses Narrative chains along with the similarity mechanism for achieving maximum efficiency. That is if the exact pair candidate event,pronoun event is not found then our system finds similar events in the narrative chains schema and makes a weighted decision Example : The drain is clogged with hair , it has to be cleaned . In this example our system first searches NC schema for pair (clog,clean-o). That is not found. One chain containing clean-o is found. Our system compares all the events in that chain with the event $clog$. One event $cover - o$ with 59.3% similarity is found. Hence our system decides answer as $clog - o$ with 59.3% confidence.	UTD's pronoun resolver just uses Narrative chains schema to search for the exact match of candidate event,pronoun event pair Example : The drain is clogged with hair , it has to be cleaned . UTD's system tries to search NC schema for pair (clog,clean-o), but it is not found hence it doesn't make any decision.
2.	Sentiment Analysis Module	

	Our system uses the same lexicon, Wilson’s subjectivity lexicon, as UTD’s system for this purpose. But the polarity assignment part is different from the UTD’s system. Our system assigns same polarity as the event, to the object of the event. This is because the action that is being done is applied on the object and the state of the object changes accordingly. Example : The city councilmen refused the demonstrators a permit because they feared violence. In the above example polarity of the event refuse is NEGATIVE. Our system assigns demonstrators to NEGATIVE entity and councilmen to POSITIVE entity.	UTD’s system assigns the same polarity as the event to the subject of the event. They assume that since subject is the one who is doing the action, he is of the same polarity as the event. Example : The city councilmen refused the demonstrators a permit because they feared violence. In the above example polarity of the event refuse is NEGATIVE. UTD’s system assigns demonstrators to POSITIVE entity and councilmen to NEGATIVE entity.
3.	Search Engine Module	
	Our search patterns use Part Of Speech from the parser to form the search pattern. This strategy exploits the sentence structure to the most. We can easily incorporate the important words and leave out stop words from the sentence. Also our search queries Q1’ and Q2’ described in the Search Engines section more powerfully capture the association of an adjective with one of the candidate answer than the other one.	UTD’s system doesn’t always form meaningful search queries like our system. Some of their search queries don’t make sense and hence can produce unwanted results. UTD’s query formation doesn’t consider the candidate answer part of the sentence and hence might miss out on some extra clue provided in the earlier part of the sentence.

	Example : The table would not fit through the doorway because it is too wide . The queries formed from our system are : Q1 : table is too wide, Q2 : doorway is too wide, Q1' : table more wide than doorway, Q2' : doorway more wide than table, Q3 : table not fit through doorway because table is too wide, Q4 : table not fit through doorway because doorway is too wide. As seen above some of the queries from UTD's system don't capture the reasoning that is provided in the sentence	Example : The table would not fit through the doorway because it is too wide . The queries formed from UTD's system are : Q1 : table is , Q2 : doorway is , Q3 : table is wide, Q4 : doorway is wide, Q5 : wide table, Q6 : wide doorway. On the other hand one drawback of our system's queries is that they are long in length.
B.	Non-common modules	
1.	ConceptNet module	
	Our ConceptNet module tries to make associations between semantic roles of candidate answers and the pronoun events or the pronoun part of the sentence. Since ConceptNet is an everyday knowledgebase it best serves the association purpose. It is able to capture more associations than UTD's associative modules. One advantage of our approach is that, since ConceptNet is a graph we can find the path lengths between various concepts which UTD's system is not able to do. The path lengths help us in determining the semantic distance between two concepts.	UTD's system uses corpus based techniques to find associations between the two concepts. They use two different techniques i.e. semantic compatibility and connective based relations to find the associations. They use some large unannotated text corpora to find such associations. UTD's system has an advantage that while searching connective based relations in text corpora they exploit the discourse connective in the sentence in a better way.

	Example : The fish ate the worm because it was hungry . Our system finds a relation in conceptNet as $hungry, MotivatedByGoal, eat$. Thus we infer that the subject of hungry is the same as subject of eat. Hence the answer is the $Fish$.	Example : The fish ate the worm because it was hungry . UTD's system searches text corpora for the relation (eat, because, hungry) if they get count more than 100 then they make further inference.
2.	Semantic role labelling module	
	For the sentences containing person names, we use FrameNet to resolve the names to the semantic roles. We have developed our own semantic role labeller. We further use the resolved semantic roles in forming search engine queries and ConceptNet queries. Example : Ann asked Mary what time the library closes, because she had forgotten . In the above sentence the semantic role labelled by our module for Ann is $speaker$ and for Mary is $Addressee$. We use these roles to form search query pattern and to find associations from ConceptNet.	UTD's system also uses FrameNet to find the roles but they just use the roles found to form the search engine query patterns. They don't use these further to find associations. Example : Ann asked Mary what time the library closes, because she had forgotten . UTD's system also uses FrameNet to find same kind of roles and form search query pattern from them.

6.2 Details about running Winograd Pronoun resolver system

1. Preparing a config file :

    - First you need to register to the website http://www.behindthename.com/api/ . This enables you to use the Named entity classifier. Then provide your key on the first line of the config file.
    - The second line of the config file is the WordNet's path on your system. Provide path of the "dict" directory from the WordNet installation on the second line of the config file.
    - Third line in the file indicates the Narrative chains schema path.
    - Fourth line indicates the FrameNet corpus path.
    - Fifth line in the file indicates the sentiment analysis corpus path.
    - Sixth line in the config file indicates whether to enable Named entity classifier or not. 0 indicates 'NO' and 1 indicates 'YES'. Provide this number on the third line of the config file. If the named entity classifier is disabled then you have to provide sentences containing resolved semantic entities in the input file. If the classifier is enabled then system assumes that you'll provide the sentences containing person names only.
    - Seventh line of the config file is the name of the Winograd schema file input to the program. The program consumes this file. Each Winograd dataset is represented in the input file by 5 lines. First line is the sentence, second line is the question. Third line contains the pronoun to be resolved. Fourth line contains candidate answers and the fifth line contains the correct answer. The new dataset will start on the new line.
    - Eighth line of the file will contain Windows Azure marketplace account key. This is required for using the Bing search API.

    Save this config file with some name and provide this as an input to the program while running it.

2. The main class of the program is QAWrapper.java. To run this class you need to provide the config file as an input.

3. Example config file :

    ```
    tej0435
    /media/F/WordNet-3.0/dict
    NarrativeChains.txt
    /media/F/FrameNet/fndata-1.5/fndata-1.5
    sentiment_analysis.txt
    ```

```
1
nyuCorpus.txt
LgzpuoIUCQCFu+0
```

4. Example input file "nyuCorpus.txt"

```
Ann asked Mary what time the library closes , but she had forgotten .
Who had forgotten ?
she
Mary,Ann
Mary

Ann asked Mary what time the library closes , because she had forgotten .
Who had forgotten ?
she
Mary,Ann
Ann
```

REFERENCES

[1] Giuseppe Pirro, Jerome Euzenat: "A Feature and Information Theoretic Framework for Semantic Similarity and Relatedness". In *Proceedings of the 9th International Semantic Web Conference (ISWC2010). LNCS 6496 Springer 2010, pp. 615-630.*

[2] Hector J. Levesque. 2011. The Winograd Schema Challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning.*

[3] Altaf Rahman and Vincent Ng. 2012. Resolving Complex Cases of Definite Pronouns: The Winograd Schema Challenge. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), Jeju Island, South Korea.*

[4] Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 787âĂŞ797.*

[5] James R. Curran, Stephen Clark, and Johan Bos (2007): Linguistically Motivated Large-Scale NLP with C&C and Boxer. *Proceedings of the ACL 2007 Demonstrations Session (ACL-07 demo), pp.33-36.*

[6] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics, pages 86-90.*

[7] Liu, H. & Singh, P. (2004) ConceptNet: A Practical Commonsense Reasoning Toolkit. BT Technology Journal, Volume 22. Kluwer Academic Publishers.

[8] Theresa Wilson, Janyce M. Wiebe, and Paul Hoffmann. 2005b. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Joint Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, pages 347-354.*

[9] Massimo Poesio and Mijail A. Kabadjov. 2004. A general-purpose, off-the-shelf anaphora resolution module: Implementation and preliminary evaluation. In *Proceedings of the 4th International Conference on Language Resources and Evaluation, pages 663-666.*

[10] Natalia N. Modjeska, Katja Markert, and Malvina Nissim. 2003. Using the web in machine learning for other-anaphora resolution. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing, pages 176-183.*

[11] Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task, pages 28-34.*

[12] Terry Winograd. 1972. Understanding Natural Language. Academic Press, Inc., New York.