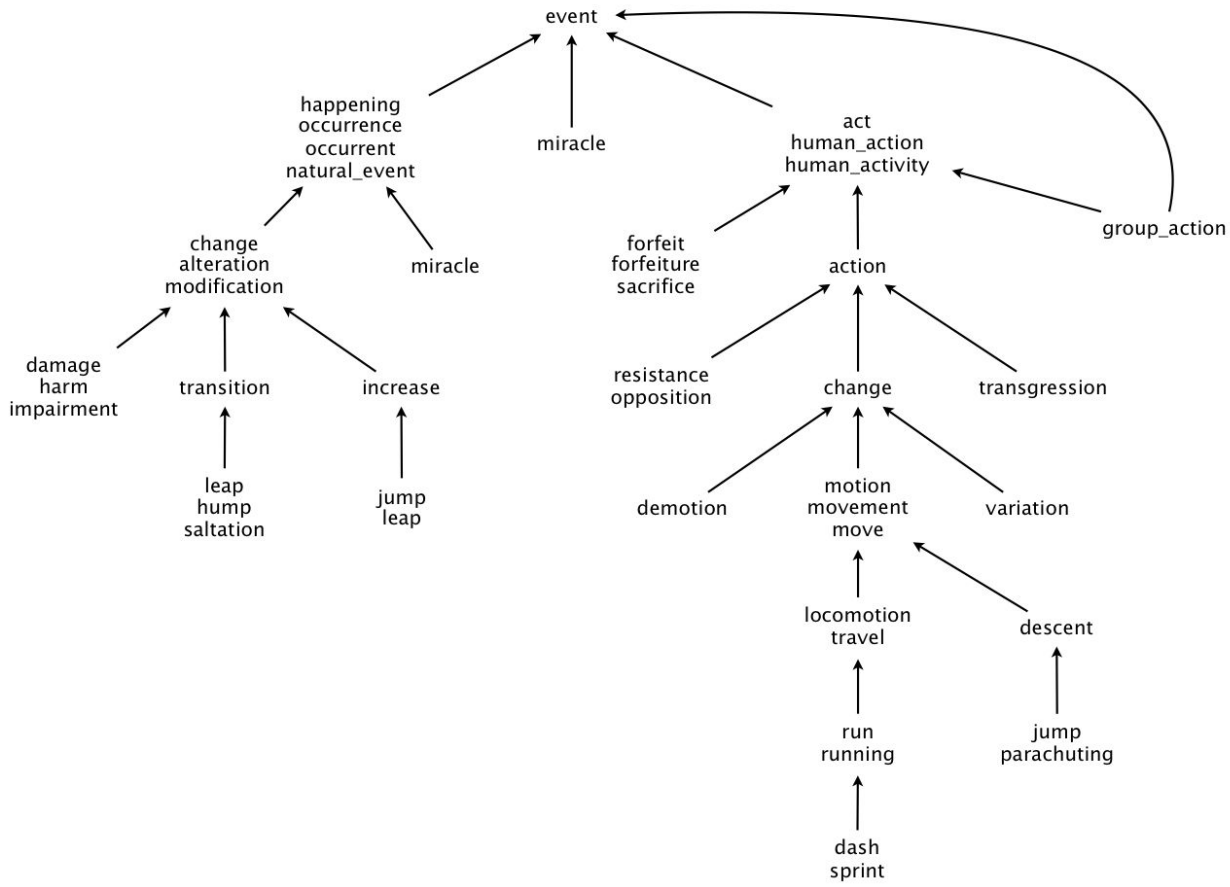
WORDNET PROJECT REPORT

INDEX:

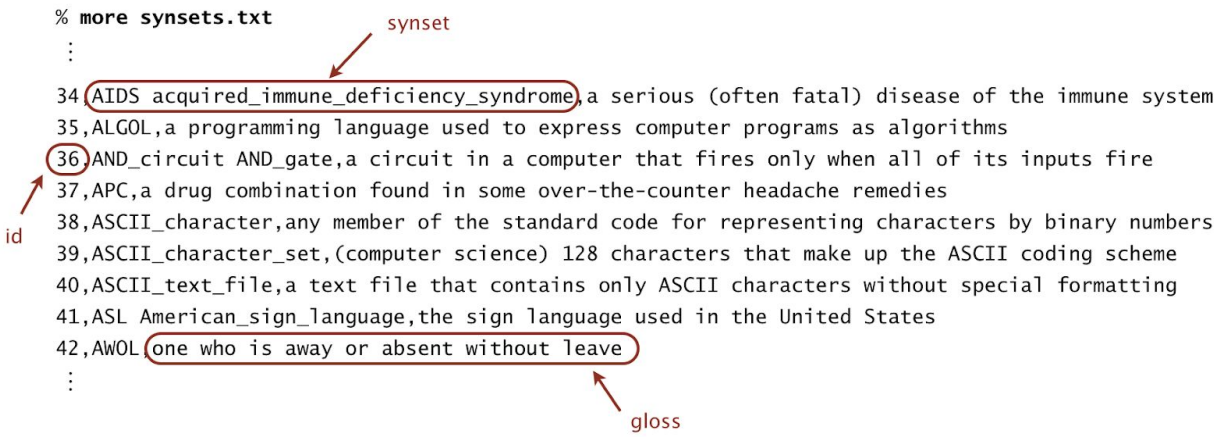
Problem Statement:

We need to build a WordNet Digraph which is one of the key components used in IBM's Watson. We have 2 word inputs called as vertices v and w which will be present in synsets. The link between v and w is called as edge that is in hypernyms.



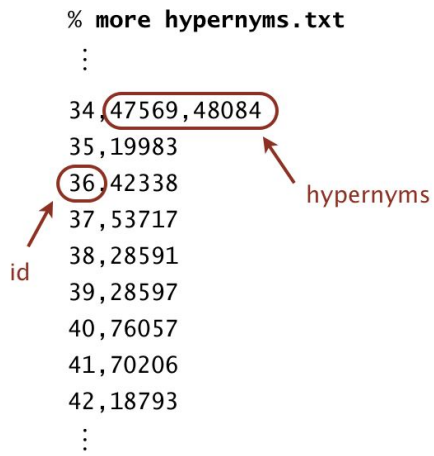
A wordnet digraph is a directed graph which traverse from child to ancestors unlike trees. So this is used in some of the applications like chatbots, word suggestions in mail and messaging applications in mobile based in the scenario that we are typing. This is built with the help of synsets and hypernyms.

Synsets:



Synsets is the set of data that contains ID, synset(noun),gloss(description) we need to create the links between these words based on the gloss.

Hypernyms:



Hypernyms are the links that maintain the data which is the link between certain words.

Related Concepts:
Some of the concepts and data structures that we have used in building the WordNet are:

- Digraph
- Bag
- HashMap
- HashSet
- Breadth First Search
- Arrays
- SAP Data Type (used to find the shortest distance between 2 nouns and common ancestors)

- WordNet Data Type (used to find distance between 2 nouns, shortest ancestral path)
- Outcast Data Type (Used to find the odd word in the given words)

Code:
Classes :

1. WordNet
2. SAP
3. Outcast

WordNet:

- WordNet(String synsets,String hypernyms) - constructor that takes filenames of synsets and hypernyms and initializes the private attributes SAP,Digraph, HashMaps noun to Id and Id to noun.
- nouns() - returns an iterable which is the set of nouns(here I have returned the keyset of noun to Id)
- isNoun(String word) - checks whether the given word is noun
- distance(String nounA,String nounB) - finds distance between the two nouns
- sap(String nounA, String NounB) - finds the common ancestor and the shortest ancestral path
- parsesynsets(String synsets) - reads the file and creates hash tables with id and nouns
- parehypernym(String hypernym) - read the file and creates the edges in digraph

SAP:

- SAP(Digraph dg) - constructor that initializes digraph
- length(int v, int w) - it is used to find the shortest ancestral path between two vertices
- ancestor(int v, int w) - it is used to find the common ancestor for v and w vertices and the shortest ancestral path.
- length(Iterable<Integer> v, Iterable<Integer> w) - it is used to find the shortest ancestral path between two vertices given v and w are set of vertices.
- ancestor((Iterable<Integer> v, Iterable<Integer> w) - it is used to find the common ancestor for v and w vertices and the shortest ancestral path given v and w are set of vertices.

OutCast:

- OutCast(WordNet wordnet) - constructor that initializes wordnet
- outcast(String [ ] nouns) - it is used to find the out cast noun among the given set of nouns

Test cases:
First, I got some compilation errors and later cleared them and faced API errors where, some of the methods and attributes were public. So, I have changed them to private and API is passed. Some of the spot bugs are there because of scanner class used. So replaced them with In class

Now, the out cast method worked perfectly but as I have used sap in wordnet both classes have logical errors. So, the ancestor output of the sap methods lead to wrong outputs as the variable that needs to be returned was returned incorrectly. Now I have changed them to the required variable and finally passed the code.

Complexities:
WordNet:

- WordNet(String synsets,String hypernyms) - O(N^2)
- nouns() - O(N)
- isNoun(String word) - O(1)
- distance(String nounA,String nounB) - O(N)
- sap(String nounA, String NounB) - O(N)
- parsesynsets(String synsets) - O(N)
- parehypernym(String hypernym) - O(N^2)

SAP:

- SAP(Digraph dg) - O(1)
- length(int v, int w) - O(N)
- ancestor(int v, int w) -O(N)
- length(Iterable<Integer> v, Iterable<Integer> w) -O(N)
- ancestor((Iterable<Integer> v, Iterable<Integer> w) - O(N)

OutCast:

- OutCast(WordNet wordnet) - O(1)
- outcast(String [ ] nouns) - O(N^2)