**CROSSWORD Generator using WordNet**

**Artificial Intelligence**

**CS 6364**

**Spring 2015**

**Nagabharan Nagendran**

**Nxn141730@utdallas.edu**

**Abstract**

WordNet®[1] is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words and concepts can be navigated with the browser. WordNet is also freely and publicly available for download. WordNet's structure makes it a useful tool for computational linguistics and natural language processing.

Crossword puzzles are riddled with facts about the world (Giant slugger Mel; The shallowest great lake; Computer that debuted in 1946), basic "language use" knowledge (Happy as ; Word after labor or arbor), not to mention basic dictionary knowledge (synonyms), plus bad puns and vicious wordplay. [2]

Crossword solving is a classic Artificial Intelligence problem. It has been described in a review paper of Computer Language Games and their role in Artificial Intelligence. In the article[3], Michael Littman finishes his description with the summary:

"All in all, Crossword Maestro is one of the most sophisticated language-game-playing programs that has been written, successfully combining a detailed lexicon, puzzle-specific knowledge, search, and even natural language explanations into a single system."

Solving crosswords feels like a uniquely human endeavor—but then again, so did chess (Deep Blue) and Jeopardy! (Watson). In the most recent American Crossword Puzzle Tournament, a computer entrant named Dr. Fill, written by Dr. Matt Ginsberg, placed in the top tier of crossword solvers in the world.[4]

The project is built using NodeJS and AngularJS in Windows environment and has dependencies on various NodeJS libraries. The project uses basic data structures in Javascript and an asynchronous design pattern for sharing information between different javascripts.

This project uses concepts of Artificial Intelligence in form of Constraint Satisfaction and Semantic ontology. The crossword is generated based on the constraint that it can use and fit most of the words given as input in the grid. Words are retrieved using the WordNet’s semantic ontology. Using the input theme we generate words using the full distance in the ontology. This project definitely has a further wide scope of exploration using WordNet’s wide semantic knowledge base.

**Introduction**

This Project builds on past work in automatic crossword generation, describing a system enhancement enabled by the availability of WordNet and other freely available software resources. To define the problem statement of this project the following criterion has to be considered:

1. Each word in WordNet is related to another through synonymy like between the world shut and close for example.
2. Words that denote the same concept and are interchangeable in many contexts are grouped into unordered sets called synsets.
3. All the synsets are linked to each other by means of a small number of conceptual relations
4. A synset also contains a brief definition or gloss for each synset member
5. Words with several distinct meanings are represented in as many distinct synsets leading to unique form-meaning pairs.
6. We extract each of these word-meaning pairs into a list
7. The list is sent to the automatic crossword generator for compiling a grid that satisfies the assignment.

The automatic crossword generation is based on a method of compiling grids based on lists of words with corresponding meanings with shared variable constraining their interlocking.

**Approach**

A very basic approach has been used in this project to solve the problem statement. We get the initial theme from the user. Using this theme we can generate the synsets containing a gloss for each. We extract the words present in the lemma of each synset and match it with its associated gloss value. Thus, each of these word-meaning pairs forms an associative array. With this array we can generate the crossword using the following algorithm:

1. Sort all the words by length, descending.
2. Take the first word and place it on the board.
3. Take the next word.
4. Search through all the words that are already on the board and see if there are any possible intersections (any common letters) with this word.
5. If there is a possible location for this word, loop through all the words that are on the board and check to see if the new word interferes.
6. If this word doesn't break the board, then place it there and go to step 3, otherwise, continue searching for a place (step 4).
7. Continue this loop until all the words are either placed or unable to be placed.

Though the above method worked it presented a quite poor crossword. A number of alterations were made such as:

1. At the end of generating a crossword, give it a score based on how many of the words were placed (the more the better), how large the board is (the smaller the better), and the ratio between height and width (the closer to 1 the better). Generate a number of crosswords and then compare their scores and choose the best one.
2. Instead of running an arbitrary number of iterations, we create as many crosswords as possible in an arbitrary amount of time. If you only have a small word list, then you'll get dozens of possible crosswords in 5 seconds. A larger crossword might only be chosen from 5-6 possibilities.
3. When placing a new word, instead of placing it immediately upon finding an acceptable location, give that word location a score based on how much it increases the size of the grid and how many intersections there are (ideally you'd want each word to be crossed by 2-3 other words). Keep track of all the positions and their scores and then choose the best one.

Thus after the crossword is generated using the above strategy we display it on a web interface along with the clues for each word.

**Experiment and Results**

The code is written in Javascript and built using NodeJS in Windows environment. AngularJS has also been installed for compilation of the code as it is dependent on the core functionality of the library. The interface is created using HTML5 and Bootstrap to give a more intuitive experience. The input is provided to the web interface and the crossword and clues are generated in their respective sections.

Figure 1 – Input Screen (Example: Sport)



Figure 2 – Crossword

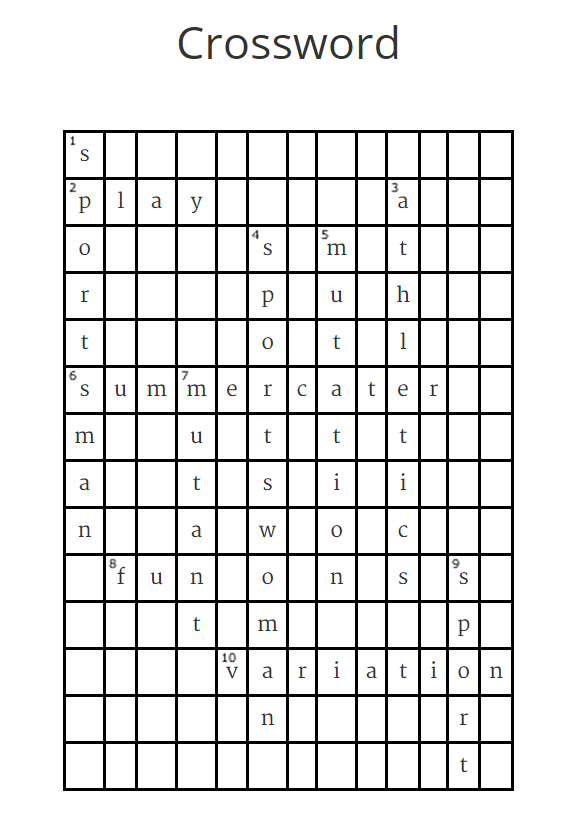


Figure 3 – Clues

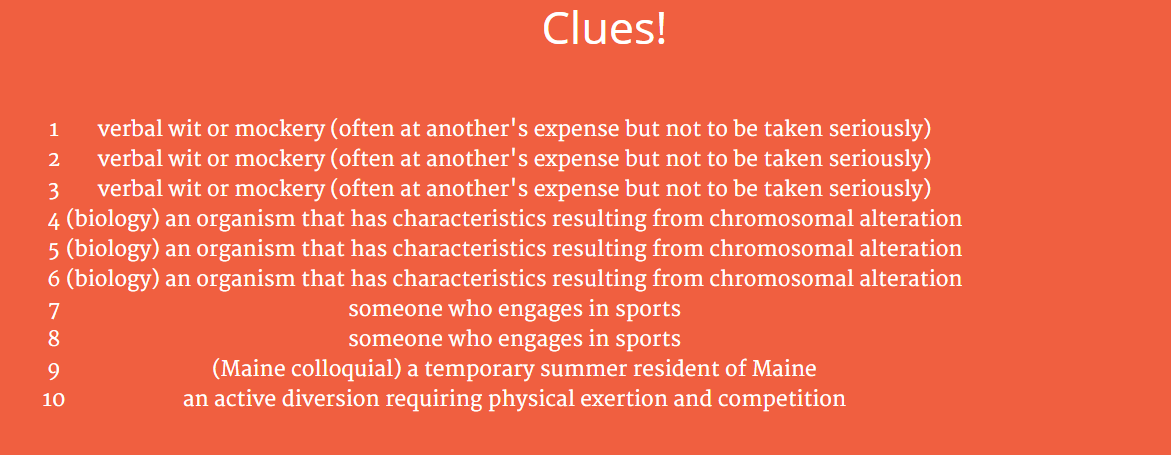


Figure 4 – Crossword for the input Earth

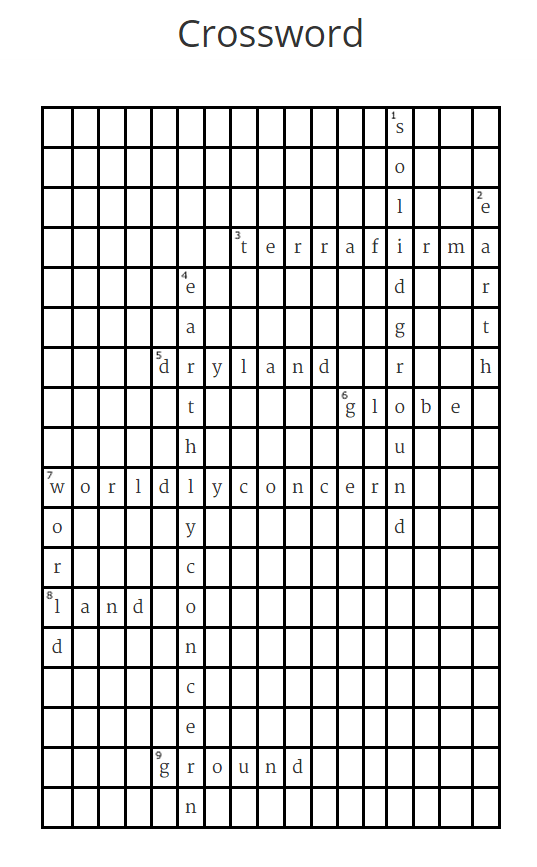
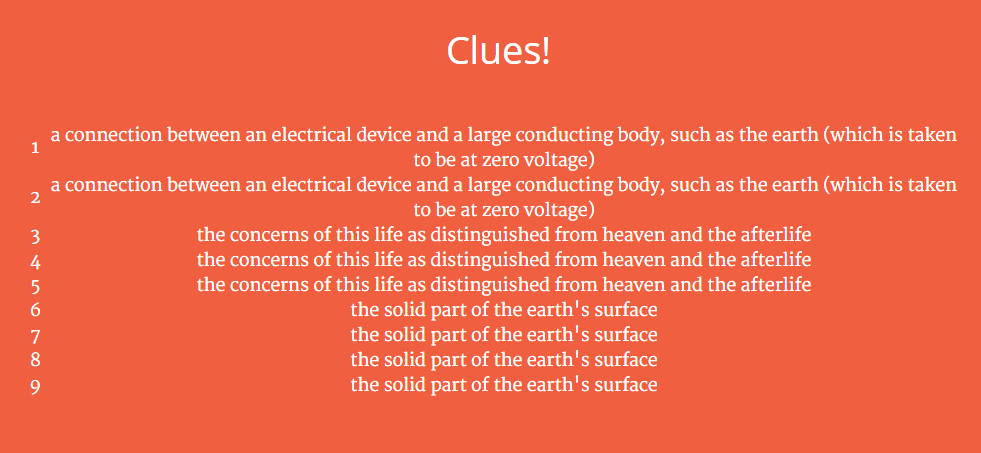


Figure 5 – Clues for Earth



**Observation**

As observed from the above screenshots we see that once input is given the crossword gets generated with words of the same semantic meaning. WordNet gives out word that are not so relevant to the primary meaning of the input word like in the case of DOG we get HOTDOG as well. We can also see that the clues generated for a group of words have same definitions. This is due to the limitation in synset where a group of lemma have the same gloss.

**Challenges**

1. Learning NodeJS, AngularJS and thier functionality
2. Designing the components of the project so that self-written code and the licensed code meets the project requirement.
3. Designing an algorithm to create a grid and fill it accordingly
4. Integrating all the components together to get the final product.

**Limitations**

Since we are restricting the synset to a particular theme we are generating same gloss definition for multiple words. If we try taking only the first input of each lemma we can’t generate crosswords in most of the instances because the synset fetch function returns the input word as part of each synset lemma. There is room for further improvement in this case.

Some words generated from the synsets though they have a semantic relationship with the theme feel irrelevant like Dog and Hotdog.

**Conclusion**

This project covered Artificial Intelligence topic of Constraint Satisfaction for crossword generation and Semantic Ontology using WordNet. NodeJS and AngularJS was used in conjunction with HTML5 to come up with the interface and logic.

As observed we can conclude that a crossword gets generated successfully based on the input theme specified by the user. There are words with same gloss definitions in some cases due to limitations in WordNet and constraint specified by applying a theme on the synset.

This project can be further explored using a different approach for crossword generation. It can also be extended to create crosswords with random words instead of themed.

**References**

1. <http://wordnet.princeton.edu/> WordNet a lexical database for English
2. <http://cs.carleton.edu/cs_comps/1213/crossword/index.php> Angry Words: An Automated Crossword-Puzzle Solver
3. Matthew L. Ginsberg. Dr. Fill: Crosswords and an Implemented Solver for Singly Weighted CSPs. Journal of Artificial Intelligence Research, 2011.
4. Michael L. Littman, Greg A. Keim, Noam M. Shazeer. A probabilistic approach to solving crossword puzzles, Artificial Intelligence, 2002.
5. Stuart Russell and Peter Norvig. Artificial Intelligence, A Modern Approach, 3rd edition., Prentice Hall, 2009.
6. George A. Miller (1995). WordNet: A Lexical Database for English.Communications of the ACM Vol. 38, No. 11: 39-41.
7. Christiane Fellbaum (1998, ed.) WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.