Construct a Database of All Word Pyramids

Isaac Lam, Dayoon Kim

May 2, 2024

Abstract

1 Introduction

This project involves the development of a comprehensive database containing all possible word pyramids. A word pyramid is a sequence of words where each subsequent word is formed by inserting a letter into the previous word, while a word chain involves changing a single letter to form the next word.

The project utilizes Dr. Z's Maple package, CRYPTOGRAM.txt, and a database of English words, ENGLISH.txt, to construct a comprehensive database of all possible word pyramids. In each pyramid, the first and last words of the pyramid are provided, and the solver must deduce the intermediate words. This follower relationship is determined by the procedure, Followers(W).

In addition to word pyramids, the project also explores another type of word sequence, where one word follows another by changing one letter (e.g., [b,a,t] -¿ [b,e,t]). This relationship is determined by the procedure, Followers1(W). The project ensures that there are no repeated sequences in the puzzles.

2 Objective

The primary goal is to compile a database of word pyramids and word chains, which will then be utilized to:

1. Create a puzzle book with challenges that provide the first and last words of a pyramid, prompting solvers to determine the intermediate words.
2. Ensure a robust system where word chains involve only single-letter substitutions without repetitions.

3 Methodology

To construct the database and develop the associated puzzle book, the project employs several Maple procedures, each tailored to handle specific aspects of word relationships:

1. Followers(W): This procedure generates a set of all words that can be formed by inserting one letter into the word W. The new word must still be a valid word in the English language as per the data provided in ENGLISH.txt. This procedure is fundamental for building the pyramid structure, where each layer represents a word formed by adding one more letter to the previous.
2. Followers1(W): Similar to 'Followers', but instead of adding a letter, it changes one letter in the original word W to form a new word. This procedure is crucial for creating chains of words where the evolution from one word to the next involves simple substitutions, which are often more challenging to solve.
3. FindAllFollowers(W, AllFollowers): A recursive function that collects all possible followers of a word W by repeatedly applying the 'Followers' procedure. It effectively maps out the entire "follower" network emanating from W, allowing us to visualize and understand the potential growth of word forms.

4. FindPyramid(Start, End): This procedure is designed to find all paths that form a pyramid from a starting word *Start* to an ending word *End*. It uses breadth-first search to explore all possible word formations incrementally by length, ensuring each word is a valid follower of the previous word.

5. Data Structure Optimization: Utilizing efficient data structures such as queues for managing the breadth-first search and tables for quick lookup, the code ensures that the search for paths and the management of word relationships are performed efficiently.

4 Implementation Details

The project leverages a combination of pre-generated connections and dynamic search algorithms:

1. Pre-generated Connections: Procedures like 'genChainConnects' and 'genTreeConnects' initialize the connection maps for chains and pyramids, respectively. These maps are critical for quick access during searches, reducing the computational overhead during live queries.

2. Dynamic Search Algorithms: Procedures such as 'findPath' and 'Reachable' are implemented to navigate through the pre-generated maps. 'findPath' focuses on finding specific paths between two words using breadth-first search, while 'Reachable' explores all accessible words from a given start point, useful for understanding the scope of possible word formations.

5 Running Examples

In this section, we will illustrate the concepts discussed in the previous sections with some running examples. These examples will help to clarify the procedures and their applications in the context of our word tree and chain puzzles.

5.1 Word Chain

Now, let's consider a word chain example with the words "fun" and "funny". Here, "bun" follows "fun" by changing the letter 'f' to 'b'. Also, "bunny" follows "funny" by doing the same thing. These form word chains as follows:

$$\{[f,u,n],[b,u,n],[d,u,n],[f,a,n],[f,e,n],[f,u,r],$$

$$[g,u,n],[h,u,n],[n,u,n],[p,u,n],[r,u,n],[s,u,n]\}$$

and

$$\{[f,u,n,n,y],[b,u,n,n,y],[f,a,n,n,y],[f,e,n,n,y],$$

$$[f,u,n,k,y],[g,u,n,n,y],[p,u,n,n,y],[r,u,n,n,y],[s,u,n,n,y]\}$$

5.2 Word Tree

Let's revisit the word "fun". Here, "faun" follows "fun" by inserting the letter 'a' between 'f' and 'u'. Also, "fauna" follows "faun" by inserting the letter 'a' after 'n'. Similarly, the word tree starting with "fun" is as follows:

$$\{[[f,u,n],[f,a,u,n],[f,a,u,n,a]]\},$$

$$\{[[f,u,n],[f,a,u,n],[f,a,u,n,s]]\},$$

$$\{[[f,u,n],[f,u,n,d],[f,o,u,n,d]]\},$$

$$\{[[f,u,n],[f,u,n,d],[f,u,n,d,i]]\},$$

$$\{[[f,u,n],[f,u,n,d],[f,u,n,d,s]]\},$$

$$\{[[f,u,n],[f,u,n,k],[f,l,u,n,k]]\},$$

$$\{[[f,u,n],[f,u,n,k],[f,u,n,k,s]]\},$$

$$\{[[f,u,n],[f,u,n,k],[f,u,n,k,y]]\},$$

Among these word trees, if you designate the finishing word as 'fauna', the solver would only provide the results of the pyramid that starts with 'fun' and ends with 'fauna', which is

$$\{[[f,u,n],[f,a,u,n],[f,a,u,n,a]]\}.$$

6 Applications and Future Work

The database and its applications in puzzle books offer both recreational and educational benefits. They serve as excellent tools for vocabulary building and understanding word formation processes. Future enhancements could include:

1. Expansion of the Word List: Incorporating more words into the database to provide richer content for puzzles.
2. Interactive Platform Development: Creating an online interactive version of the puzzle book for broader accessibility.
3. Integration with Other Linguistic Tools: Linking the database with other linguistic analysis tools to provide deeper insights into language structure and evolution.

7 Conclusion

The construction of a database for all word pyramids, supported by efficient algorithmic procedures, provides a valuable resource for linguistic exploration and educational entertainment. This project not only advances our understanding of word relationships in the English language but also offers a fun and engaging way for users to interact with language learning. The methodologies and technologies used highlight the project's innovative approach to solving complex problems in linguistics.