Brian Pinson
U91813366
Automata Final Project
11/14/2017

12. Computational Linguistics – FSM Lexicon

Introduction

In linguistic theory it is believed human language consists of two parts a lexicon and a grammar. The lexicon is a catalog of words in a language while the grammar is the rules responsible for the language. Morphology is the study of words, the composition of words, and how they relate to language. They can decompose a word into its root, prefixes, and suffixes. Finite State Morphology (FSM) was originally invented for the analysis and synthesis of the Finnish language. Since then it has expanded to model many other languages. FSM breaks down language into rules and pieces that can describe language as a finite state machine. The FSM uses a lexicon structure and a set of rules as its lexical transducer to model a natural language.

A simple morphology using FSM could consist of adding letters to form a word, for example: d+o+g. The problem is that simple morphology couldn't express irregular plural nouns like goose and geese. To exist is to be but the past tense of be is was and the past participle is been. Derivational morphology uses derivation to build words by adding prefixes and suffixes to a root word. An example of a lexical transducer is creating the word ate with eat+Past or eaten with the word eat+PastParticiple.

Task

The project was designed to create a Finite State Morpology (FSM) Lexicon that would allow the user to construct words using morphemes. Morphemes would act as the lexical transducer building upon themselves to create a variety of words with completely different meanings. The easiest way to design a lexicon where the user would be most familiar with the morphemes they were working with was to use English words constructed from Latin or Greek. For example any word containing "octa" implies it has eight of some attribute. For example an octagon is a polygon with eight sides, an octopus has eight arms, and octoechos is a hymn with eight parts. When looking at words with Latin or Greek roots it is easy to estimate the meaning of a word even if they have never seen it before.

The program asks for a text file's name and will take the information to build the lexicon. The lexical entries include the morpheme, lexical transducer, and child node names which are morphemes that build upon the lexical entry. The lexical entry must take the child node names and find where they are stored in the lexicon so that they may be connected to their parent node.

The format of the input file should follow this format at every new line:
transducer.morpheme n child_t1.child_m1 child_t2.child_m2 … child_tn.child_mn
Where n is an integer, the value of n should be the number of connected child nodes.

The FSM must first ask the user to determine the initial state. It must also confirm that the initial state exists in the lexicon. If it doesn't the user will be asked to attempt again to name a morpheme that exist somewhere in the lexicon. Starting from the initial state all child nodes are displayed. The user inputs which child node they wish to build their word with by picking the lexical transducer. The child node then becomes the new present state. If that state is end or the child node has no children node of its own then then the FSM has reached a final state and FSM() terminates. Otherwise the process continues and the new node displays all available child nodes. When FSM() terminates the user can construct a new word with lexicon or quit by typing "quit" when choosing the initial state.

Results

The program is successfully able to build a FSM Lexicon that a user can navigate and construct words from morphemes. It is easy to navigate and user friendly to operate. The program not only constructs the word but constructs the word's definition using information about the morphemes. As long as the input is correctly formatted it can handle thousands of randomly generated inputs. Beyond morphemes the program can handle single letters or even symbols as long as the formatting of the input file is correct.

Conclusion

A FSM lexicon is able to store exponentially more words than a regular dictionary with significantly smaller file sizes just by storing morphemes as its lexical entries. The words and meanings are constructed through the user operating a finite state machine. Some problems that could occur is with irregulars in English. Using FSM the word mouse can't be changed to its plural form unless it is

completely replaced. This would require having a large list of all known irregular words and having the lexicon check each time the FSM moves to a new state. This can hamper the efficiency of the lexicon significantly.

Discussion

FSM lexicons double as type of data compression. For instance chemical nomeclature is a set of rules to construct names of chemicals and compound. FSM could be used to direct the naming processes while the build blocks of chemical names could be stored as lexical entries. Many other scientific naming systems often reuse words of Latin origin and could be broken down into morphemes. Information about a species can be surmised just by translating the Genus-species name given to an organism. There is a lot of potential in the scientific field which have strict naming procedures for constructing and decomposing words into smaller discrete pieces.

Discoveries

FSM lexicon program was very stable with few surprises. The field of FSM lexicons has much more room for development. There are so few books and journals taking advantage of this system it feels underused and still unexplored.

Bibliography

Hanks, Patrick. *Lexical analysis: norms and exploitations*. MIT Press, 2013.

Kenneth R. Beesley & Lauri Karttunen, Finite State Morphology. Stanford, CA: CSLI Publications (distributed by the University of Chicago Press), 2003. xviii + 505pp. And CD-ROM. ISBN hardbound 1-57586-433-9, paperbound 1-57586-434-7