Computer Linguistics and Sentence Synthesis

New Mexico
Supercomputing Challenge
Final Report
April 3, 2013

Team # 51
Los Alamos High School

By:
Connor Bailey and Hayden Walker

Teacher Sponsor:
Mr. Lee Goodwin

Mentor:
Mr. Rob Cunningham

Table of Contents

Executive Summary

In our supercomputing challenge project, we wrote a program in the Python programming language that "learns" new words of the English language and applies those words in a simple sentence. The program builds a sentence so that we can easily tell if the program has gathered enough information to properly use the word. This is how we are able to tell if the program has learnt the word or not. If the program builds a sentence that is unintelligible, then we will be able to tell that right away. The program learns new words in a way similar to how the human brain does. The human brain is a very complex and powerful system, but we have simplified the process down to how it actually learns new words. We applied our simplified method to our Python program with the help of our own code and the natural language toolkit. Our program is able "learn" new words. It is taught words one word at a time. Then the program stores those words in a dictionary. It keeps all of the information it has on the word in the dictionary as well. It uses the context of where the word is in the sentence, the ending of the word, and help from NLTK tagging in order to do this.

Our program has huge potential of reaching further into the difficult areas of natural language processing, computer linguistics, and Artificial Intelligence. Our program proves that at least on a small scale, a computer can be "taught" how to learn a word. It can then demonstrate that it has done so by writing a sentence using the word. We are able to see whether or not it was a success by just reading the sentence as we have a full working knowledge of the English language.

Problem Statement

We have attempted to take on the problem of creating a program that is capable of understanding an English word. We have created a setup in which the program is able to show that it knows the word. And it is easy for us to determine if the program used the word properly because we are fluent in the English language.

We wanted to look into this problem because we have observed the difficulties involved in the understanding of a language. We have also seen the incredible abilities that people would gain if computers were able to correctly understand human language. We then realized that simple sentence synthesis would be the first step in that process. It was for these reasons that we decided to begin working on this project.

Methods

In our project, we assembled a Python program that "learns" and creates sentences. The program first builds a dictionary of words that it is taught. It then gathers all of the information that it can about that word in the value portion of the dictionary. This information can include, part of speech, word ending, use in previous sentences, and modifiers of the word. It then attempts to build simple sentences using this information about the words. The sentences are built using knowledge of the word or other word specific attributes that our program can detect. These specific attributes would be things like word ending, which can be used to help the program identify how the word can be used. All of this is stored in the dictionary and can be used for helping the program properly "learn" to utilize them in a sentence.

When the program constructs the sentence, it first begins with a subject and a verb. Currently, our program chooses the word randomly. But later, if expanded on, the program could be made to choose the word intelligently in response to a human. The next thing that the program does is choose a verb. It also chooses this verb randomly at first. But, then, the program checks to make sure that the words are compatible. It uses all of the information that it has stored to determine the compatibility of the words. Then, once this has been done, if the word is incompatible, it will choose a new word until it is compatible.

Once it has found a compatible subject-verb sentence, it begins to add complexity to the sentence. Once again, it does this randomly but could be expanded in the future.

Materials

We used Python 3.1 to write our program. We also use the natural language toolkit (NLTK) for Python in our program.

Results

We accomplished the task of creating a system in which we are able to both teach and test a Python computer program words, sentences, and other grammatical structures in the English language. Our program can be taught words by a user, then it stores the information that it has learned on that word. It is then able to create sentences that it deems to be correct based on what it knows.

Our program represents huge opportunities in pushing the boundaries in fields such as computer linguistics, artificial intelligence, and natural language processing. Our program deals with all of these things and is working proof that it is possible for a computer to properly use a word in a sentence. This is as close to proving that the computer understands the word as it is possible to get. In the world of education, it is commonly proof that one understands a word by showing that they can use it correctly in a sentence. This is commonly used on standardized tests among other things. This is why we chose for our program to demonstrate it's knowledge in this way. We also did this so that we would be able to be able to immediately determine whether or not the computer understood the word. With this setup, we are able to do that just by reading the sentence that the program produces.

Personal Statement

In our project, we managed to create a program that can generate simple subject-verb sentences that make sense and use all of the words that the program has "learned" properly. This is exciting because it shows that this type of thing can be built on to the point where computers are able to build and use any sentence in the English language. It is also exciting that we are able to show that a computer can effectively “understand” a word through storing information on that word and applying that to making a sentence using that word.

Future Work

In the future, it would be a good idea to expand on this project so that the program was able to generate more and more complex sentences. This becomes exponentially more difficult but is also a lot more rewarding. Another good way to expand this project would be to make the computer generate the sentences based on something specific like replying to a conversation or informing a user of an event. This is where a lot of AI would come into play. This would make it infinitely more intuitive for a user to use their computer as the computer would be able to translate into English what it wished to inform the user.

It would also provide an interesting view if this program were expanded on to enable it to work in other languages besides English as well. It could then be applied as a translation tool for helping switch between languages. By being able to actually understand the word, the translator would be at a much higher level than any translation software available today.

Our program could be implemented in any environment in which a human interacts with a computer.   Our program has the specific goal of eventually being able to make interactions between people and computers seamless as they will both be “speaking” the same language. IT could also be used to make other computer or robot processes run even more smoothly.

Acknowledgements

We would like to thank our mentor Mr. Robert Cunningham for aiding us on all of the steps throughout the process of developing our project. We would also like to thank Drew Einhorn for reviewing our proposal and providing helpful feedback that we were able to use in our project. We would also like to thank Supercomputing Challenge volunteers especially those who judged during the evaluations.

Bibliography

"Computer Learns Language by Playing Games." MIT's News Office. N.p., n.d. Web. 02 Apr. 2013.

"Dark Times for Solar." MIT Technology Review. N.p., n.d. Web. 02 Apr. 2013.

"How the Brain Learns: The Blog." How the Brain Learns The Blog. N.p., n.d. Web. 02 Apr. 2013.

"Language and the Brain." Language and the Brain. N.p., n.d. Web. 02 Apr. 2013.

Loper, Edward, and Steven Bird. NLTK: The Natural Language Toolkit. [S.l.]: [S.n.], [..]. Print.

"Natural Language Toolkit¶." Natural Language Toolkit — NLTK 2.0 Documentation. N.p., n.d. Web. 02 Apr. 2013.

"Preface." Computational Linguistics: Models, Resources, Applications. N.p., n.d. Web. 02 Apr. 2013.

"Reading Trouble May Start with the Brain Having Trouble Processing How the Words Are Taught to the Student." DailyRx. N.p., n.d. Web. 02 Apr. 2013.

Glossary

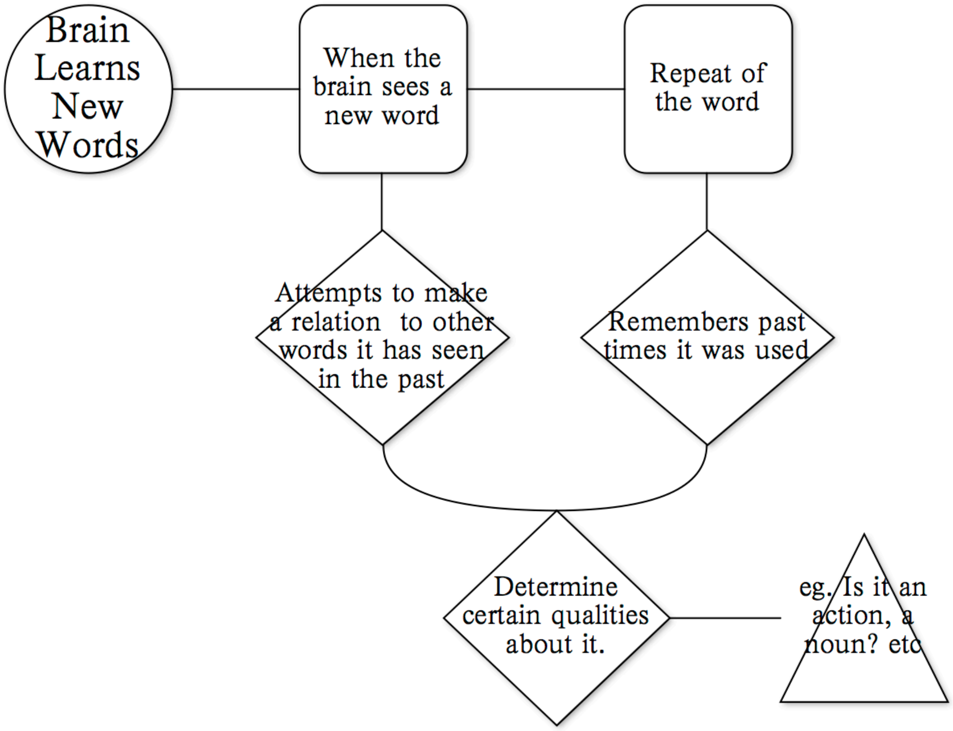
NLTK: Natural language toolkit. We used a pre -written set of code for natural language processing with Python to help us determine words quicker in our program.

Computer Linguistics: Computers trying to "understand" human languages.

AI: Artificial Intelligence. Intelligent life faked by a computer system.

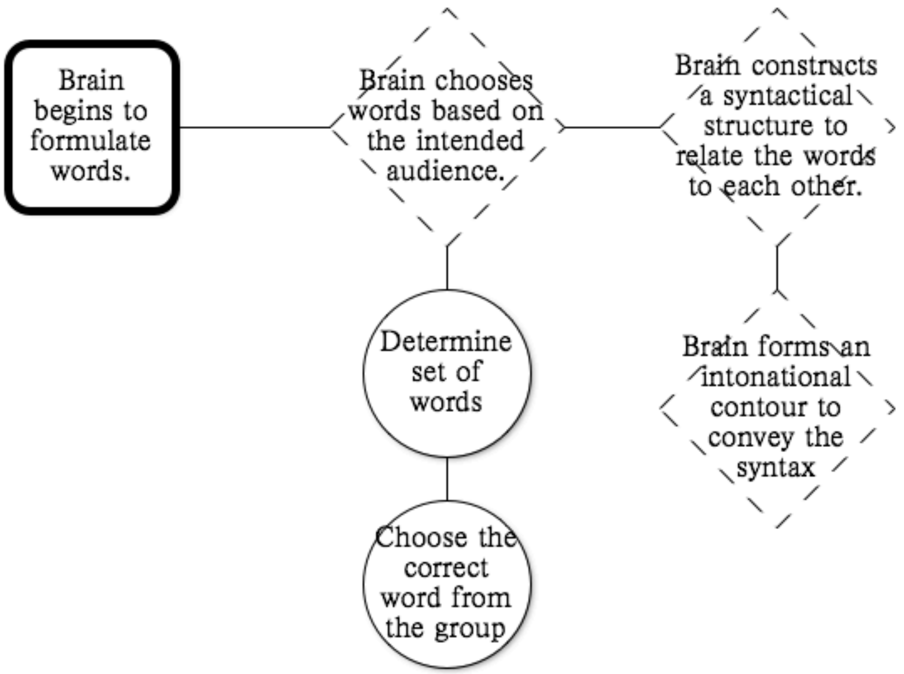
Appendices

Appendix I: The simplified process of what happens when the brain sees a new word.


Brain Learns New Words
When the brain sees a new word
Repeat of the word
Attempts to make a relation to other words it has seen in the past
Remembers past times it was used
Determine certain qualities about it.
eg. Is it an action, a noun? etc

This is what we modeled our program to do when it sees a new word as well.

Appendix II: Brain begins to formulate words.


Brain begins to formulate words.
Brain chooses words based on the intended audience.
Brain constructs a syntactical structure to relate the words to each other.
Determine set of words
Brain forms an intonational contour to convey the syntax
Choose the correct word from the group

http://www.science20.com/news_releases/how_our_brain_chooses_right_words
http://www.hms.harvard.edu/hmni/On_The_Brain/Volume04/Number4/F95Lang.html

This is the process that the human brain goes through when it begins to attempt to formulate a sentence. We simplified this process down, and attempted to mirror it in our program.