Can Natural Language Processors Help Unlock the Black Box of Language Comprehension?

By

Tyree S. Cowell

A Thesis Submitted in Partial Fulfillment of the

Requirements for the Degree of Bachelor of Science

With Honors in Cognitive Science from the

University of Michigan

2018

Advisor: Dr. Jonathan Brennan

Abstract

One of the most overwhelming challenges facing researchers in Cognitive Science today is the mystery of how the brain parses and comprehends language. Studies in Computer Science are plagued by a similar challenge: how to create a program that can effectively process the grammar of human languages. Through this thesis, I hope to combine these two problems by comparing certain outputs of Google's SyntaxNet to EEG data of subjects listening to naturalistic language data, more specifically, Alice in Wonderland text. The Natural Language Processor I will be analyzing (SyntaxNet) processes each sentence incrementally. At each new state of the parser, it decides to perform one of the following actions: make a right or left arc between grammatically associated words, or shift another word onto the stack. Through some manipulation of the SyntaxNet code, as well as an original program, I extracted the timing and frequency of these decisions, pairing them with each word they were based on. Similarly, the EEG data collected by Dr. Brennan's Lab at the University of Michigan couples brain signals with the time at which a subject heard a specific word. Cross-referencing these two data sets allowed us to test the hypothesis that the parser's actions correspond, in some way, to cognitive actions that humans use to understand sentences. Our results showed that there is a correlation between the changing stack depth of the parser and positive frontal activation in the brain. There is also a correlation between number of right arcs and late negative activation. While these results have a number of possible causes, they support the hypothesis that natural language parsers may be useful in providing a plausible model for the black box of language comprehension in the human brain.

Can Natural Language Processors Help Unlock the Black Box of Language Comprehension?

Language processing in the brain has often been referred to as a "black box problem". The term was first coined by Noam Chomsky, the father of modern linguistics, in reference to the process of language acquisition and the "insufficient stimuli" conundrum. Linguists realized that while each child receives vastly different language input, they all form essentially the same internal grammars. Additionally, as language research progressed, researchers came to realize that there was no obvious way to observe the language processes that occur in the brain. In both of these situations, the input and output of language processing are accessible, but the internal mechanisms and processes are nearly impossible to observe. This is why the term "black box" is so accurate. Scientists have spent years attempting to get a glimpse into this black box, creating many innovating techniques in the process.

Recent work has made progress on this problem by pairing natural language processors with behavioral and neuroscientific data. According to Hale et. al. (under review), this methodology has been applied to several NLP program so far, including Long Short Term Memory (LSTM) networks, which did not perform very well, and Recurrent Neural Network Grammar (RNNG) models, which performed decently well. (Here performance is measured by the existence of correlations between the NLP and neuroscientific data). The current study is a subset of this overarching investigation with a focus on Google's natural language processor, SyntaxNet, and its parser, Parsey McParseface. Due to the previously mentioned success with RNNG models, we suspect that SyntaxNet will correlate reasonably well with EEG data of participants listening to naturalistic language data. This is because, both RNNG models and SyntaxNet operate on the overall structure of building a phrase structure tree using a symbolic

stack. This structure mimics the popular theory that humans process language incrementally, and form complex, nested internal phrase structure trees in order to understand the meanings of utterances. (Abney & Johnson, 1991) (Brennan & Hale, 2017) (Nelson et. al. 2017)

In order to discern if SyntaxNet is a cognitively plausible model for the brain's language parsing mechanisms, we must first understand how SyntaxNet works. This understanding is also crucial if we hope to make any sense of the results we gather. The following section, describes in a slightly abstracted manner how SyntaxNet and its parser, Parsey McParseface, produces a dependency tree based off a sentence as input.

SyntaxNet/DRAGNN:

SyntaxNet is "a globally normalized transition-based neural network model that achieves state-of-the-art part-of-speech tagging, dependency parsing and sentence compression results" (Andor, 2016). It went open-source in 2016 and it is the most accurate language processor to date, boasting a 94% success rate on well-formed English text and a success rate just over 90% on ill-formed English sentences. In other words, Google has designed a tool that nearly reaches human standards of accurate language parsing (around 96-97%) (Petrov, 2016). This tool, along with many other tools like it, has been very useful in engineering settings as well as research into artificial intelligence and computer science, but it has rarely been applied to the study of cognitive neuroscience or, more specifically, the problem of natural language processing.

The easiest way to understand SyntaxNet is to first understand what a DRAGNN is. DRAGNN, a model also created by Google, stands for Dynamic Recurrent Acyclic Graphical Neural Network. According to Kong et. al., (2017) a DRAGNN is not a particular neural architecture but rather a formulation for compactly describing neural architectures. SyntaxNet's parser, Parsey McParseface, is an ideal application of this model. Essentially, a DRAGNN is a

system built of connected Transition Based Recurrent Units (TBRUs). A TBRU is made up of several components or functions that help it achieve an overall goal:

- A transition system, T
- an input function, m(s)
- a recurrence function r(s)
- An RNN (recurrent neural network) cell.

A transition system, simply put, uses a transition function to take the current state of the system, compute what action to take based on the current state, and return a new state based on the action it chose. The input function m(s) takes in raw information about the current state, and translates it into a format that the system can understand. A recurrence function connects the current state with previous related states, and the RNN cell computes a hidden representation (h(s)) which is later used by the transition function. Figure 1 gives a general idea of the framework of a TBRU.

Parsey McParseface can easily be described in the context of a DRAGNN. It uses an arc-standard transition system (T). This means that the current state is determined by the stack (previously described), the collection of potentially correct parse trees that Parsey has constructed up to that point, and the list of words that the system has not yet interpreted. The developers of SyntaxNet call this collection of information the *configuration* of the parser. The transition system uses this configuration, to decide which step to take next. After taking that step, it returns the new state of the configuration. This process repeats until the program has a fully formed parse tree for the given sentence.

But how does Parsey decide which words are related to each other? First, it runs the sentence through a part of speech (POS) tagger, which uses a feed-forward neural network

classifier to extract specific features from a given word (i.e. gender, number, case, etc.) in order to determine its part of speech. This gives the system a vague idea of the role of each word in the sentence. Once this representation has been generated it gets passed to the parser. Parsey uses a dependency parser model which "uses an approach to sentence structure based on the idea that one word "governs" another, potentially at any distance" (Kong et. al, 2017). This parser moves left to right, analyzing each word individually and slowly building a parse tree identifying the relations between each of the words in the sentence. After looking at a new word, there are 3 decisions the parser can make: SHIFT, LEFT_ARC, or RIGHT_ARC. A SHIFT means that the parser cannot make any connections based on its current configuration, and it wants to take in the next word for more information (thus it 'shifts' to the next word). An ARC means that the parser has found a relationship and is connecting two words. Arcs are always formed from head to modifier, the direction (LEFT or RIGHT) indicates which direction the arc is going (i.e. if the modifier is before or after the head).

Neural Correlates:

The main goal of this project is to ascertain if there are any correlations between the actions taken by the parser, and the actions presumably taken by the human brain during language processing. Our first hypothesis poses that there will be a correlation between stack depth as a measure of the parser's memory load, and the working memory load of participants listening to naturalistic language data. It has long been accepted that working memory plays an important role in language processing. According to Cowan (2017), it is impossible to comprehend words within conversation or text without remembering necessary background information given by previous words or sentences. Additionally, Cowan observed in his 2017 study that individuals with memory deficits seem to struggle more than healthy individuals with

sentences that require recitation or re-processing. Importantly, if we observe a correlation between the stack depth of the parser and the memory load of the participant, it will indicate that the ways in which SyntaxNet handles memory is a cognitively plausible model for how human memory functions during language processing.

Our other generalizable hypothesis is that we expect to see a correlation between the number of arcs the parser makes per word, and a measure of the structure building load the brain takes on during language processing. This hypothesis is supported by the work done in Abney & Johnson (1991) where they correlated right and left arcs to a "node count", representing how many nodes the participant added to their internal structure. This hypothesis is based on the widely studied and previously mentioned idea that humans process language incrementally, and that they build internal syntactic structures in order to understand the meaning of utterances. (Abney & Johnson, 1991) (Brennan & Hale, 2017) (Nelson et. al. 2017). If we do find some kind of correlation between the number and timing of arc creation in the parser, it may indicate that SyntaxNet builds internal structures in a similar fashion to the human mind.

Method

SyntaxNet Data:

In identifying which potentially applicable quantitative data could be extracted from SyntaxNet, we performed an analysis of the inner workings of SyntaxNet and the associated DRAGNN framework. After completing this analysis, it made sense to extract the frequency and timing of the decisions the parser makes as it works its way through a sentence (i.e. SHIFT, LEFT_ARC, or RIGHT_ARC-- described above). Using the information provided on the GitHub repository for SyntaxNet and DRAGNN, I was able to run the DRAGNN docker through an

online Jupyter notebook, using a free trial of the Google Cloud Platform. (Wu, 2017). This program provided two options upon providing an input sentence: receive a dynamic trace explorer tree outlining all the steps the parser took, or receive a well-illustrated parse tree. While the trace explorer did help to provide a better understand the inner-workings of DRAGNN, the parse tree output was better aligned with the goals of this project. The parser was made to process every sentence in the Alice_text data set that was used in the corresponding research study (Brennan, J. R. & Hale, J. T. (Under review)), meaning that these sentences were the exact sentences heard by the participants being recorded with EEG. (see Appendix A for full list of sentences)

Within some manipulation of the Jupyter code, the program printed out all the decisions made by the parser (SHIFT, LEFT_ARC, or RIGHT_ARC), as well as the configuration of the parser upon which the decisions were based. This output was converted into one text file per sentence. Each text file consists of a list of numbers correlating to the decision made by the parser. A ‘1’ correlates to a SHIFT, a ‘2’ correlates to a LEFT_ARC, and a ‘3’ correlates to a RIGHT_ARC. Understandably, these numbers remained in the same order as the decisions made by the parser. My own program (see Appendix 2 for full code) took these files as input as well as the file containing all the sentences of the Alice text. The program essentially goes through the parser decisions for a given sentence one by one and associates each decision with a word in the sentence, using SHIFT decisions to advance to the next word. The final result of this program is a table stating which decisions (and how many of each decision) the parser made after taking in a given word of the Alice text. Table 1 is an example of what this table would look like for a single sentence of Alice text.

There are several important things to note on the example table. First is that it demonstrates the relationship between arcs and stack depth. The overall formula for stack depth is as follows:

$$\text{Stack depth} = \sum \text{words added to stack} - \sum \text{arcs up to that point}$$

You can see on the chart that once 'Would' is added, the stack depth goes up to 1. This trend continues until the word 'never' is added. Once the processor adds 'never' to the stack (making the stack depth = 4) it is able to make a left arc. This left arc decreases the stack by 1, reducing it back to a total of 3. Another important note is that this chart exhibits a trend that persists throughout all the sentences: Right Arcs tend to occur collectively at the end of the language processing time-frame. This fact will be important later when we look at results.

Once this type of data had been collected for every sentence in the first chapter of Alice in Wonderland, it was time to combine it with the EEG Data collected in the corresponding research study (Brennan, J. R. & Hale, J. T. (Under review)). The following section gives more information about how the EEG data was collected prior to my work on this project. It then goes on to discuss the statistical methods used to collate the two data sets.

EEG Data:

The EEG data collected by Dr. Brennan's lab was collected and cleaned in a series of steps. Data was collected from 33 participants and recorded at 500 Hz. Participants were asked to passively listen to a 12.6 minute audiobook consisting of the first chapter of Alice in Wonderland by Lewis Carroll. The recording from the EEG was matched up with the start time of the audio, and the data was epoched to -0.3 - 1.0 seconds surrounding each content word the participant heard. The raw data contained recordings for 2129 words. Ocular signals (artifacts caused by eye-blinks) were removed using ICA and other noise artifacts were removed manually. Data

processing is describe in more detail elsewhere (Brennan & Hale, Under Review). After the data was cleaned, there remained data for about 600-800 words per participant, each with recordings from 61 channels distributed across the scalp. In other words, the result of this process was a 3D array (61 channels by 600-800 words by 500 time points) of data for each of the 33 participants. After organizing the two main data sets (SyntaxNet and EEG), simple statistical methods were used to combine them. The following section outlines these methods.

Statistical Analysis:

The flowchart in Figure 2 portrays how all of the data was combined to calculate the results we found. On the left side, you can see the process described in the previous section, by which the raw EEG recordings were aligned with the auditory stimulus (the audio book), segmented into epochs surrounding each content word, and cleaned. The right side of the chart explains the collection of the SyntaxNet data, by which we gave the story text to Parsey McParseface and combined it's output with the time aligned words, giving us the example chart shown in Table 1, containing per-word stack depths, arc actions and covariates.

Once we had this information we used a simple linear regression to estimate a linear function between word-by-word parser states and the EEG data. We also performed this analysis on a random set of parser states, obtained by permuting the actual parser states, in order to construct a proper null model. Next, we tested for which channels and for which points in time across the entire group did the actual parser perform better than the control states. The meaning of "cluster-based permutation test" on the chart is simply that we wanted to obtain results where there was a cluster of data points that significantly differed from the null model, rather than just individual data points that could be outliers. This step followed the procedure described by Maris

& Oostenveld (2007). The following sections explains the results we found and what they could mean for the overall goal of this project.

Results

After running a simple linear regressions and discerning when the parser produced a correlation that was significantly stronger than that of the null model, we gathered the results portrayed in Figure 3 and Figure 4. We used a p-value of < 0.05 to determine significance in both graphs. Figure 3 demonstrates that words associated with deeper stacks (i.e. a larger memory load) are correlated with more positive voltages on frontal electrodes at around 600 - 700 milliseconds after word onset. We found this correlation to be significant with a p-value of 0.008. Similarly, in Figure 4, we can see that words associated with the creation of more right arcs are correlated with more negative voltages in the frontal region at around 700 - 800 milliseconds after word onset. We determined this correlation to be marginally significant after calculating a p-value of 0.047.

Of all the metrics shown in Table 1, the only one that did not demonstrate a significant correlation to the EEG signals was the number of left-arcs per word. There was a correlation with significance of $p = 0.05$ to signals starting at 0 ms. However, we deemed this result to be insignificant as there is unlikely to be any language processing occurring at 0 ms since the subject would not even have finished listening to the content word at that time. This result is likely an artifact of the data and not a relevant finding. The following section, summarizes some of the possible implications of these results, especially as how they relate to the original hypothesis correlating stack depth to memory load, and occurrence of arcs to a structure building metric within the brain.

Discussion

First focusing on the results associated with stack depth, we notice that this intuitively makes sense. As mentioned in the introduction to this paper, it is widely accepted that working memory is essential to language processing. Figure 3 shows that there is a significant correlation between the timing of SyntaxNet's memory usage, and frontal lobe activity in the brains of our participants. Many studies including Fletch & Henson (2001), conclude that various regions in the frontal cortex are activated as a result of memory processing. Given this information it is reasonable to conclude that the correlation shown indicates a relationship between the memory usage of the parser and of the working memory load in the brain.

This information could indicate a lot about the way humans process language, simply because we understand the inner-workings of SyntaxNet. More specifically, SyntaxNet only adds to the stack when no more decisions can be made about various words' roles in the sentence, and only deletes from the stack when an arc can be made and a role can be assigned. If the human brain uses memory in the same way, that would connect with theories that the brain parses language incrementally, as opposed to all at once. Furthermore, this conclusion is supported by several previous studies, including Nelson & Dahaene (2017) who observed an increase in high gamma power after each additional content word that the subjects hear, as well as a sudden decrease in the same metric whenever words could be merged into a phrase. This pattern mirrors the changing stack depth of Parsey McParseface as it processes a sentence.

The results associated with right arcs shown in Figure 4, showing a correlation between occurrence of right arcs and negative, late frontal activation, are a little more difficult to understand. This difficulty is mainly caused by the fact that the role of "arcs" in the brain's parsing process is still under investigation. While most linguistic theories propose that the brain

constructs a structure of nested phrases, the cognitive neuroscience community still does not have a lot of answers with regard to how and whether the brain does this (Nelson & Dehaene, 2017). However, one potential explanation for the occurrence of right arcs can also be found in Nelson & Dehaene (2017).

As mentioned in the Method section of this paper, while left arcs occur throughout the parsing process, right arcs tend to collate at the end of the sentence (after the last word is added to the stack). Nelson & Dehaene explain an "end of sentence effect", meaning that there must be a series of structure building actions taken at the end of a sentence so the brain can construct an accurate representation of the utterance as a whole. In their 2017 study, they observed a strong activation in both frontal and temporal areas towards sentence endings in proportion to the amount of information being processes. This is very similar to the occurrence of right arcs, which tend to increase with the number of words in the sentence.

Finally, it is important to discuss the fact that no significant correlations was found between the EEG data and the occurrence of left arcs within the time window that aligns with sentence processing (e.g. >200ms after word onset). So while it is possible that this finding is merely a coincidence, or an artifact of the layout of the data, more research is needed to confirm beliefs one way or the other.

While the findings of this research are exciting, these results are by no means a finished product. It is clear that more research is needed on both sides of the cognitive neuroscience - computational linguistics relationship. If neuroscientists can give a better picture of the structure building processes in the brain, it will be easier to link them to mechanisms in natural language processors. Similarly, if work in computational linguistics continues to improve the accuracy and

performance of natural language processors, they will be more helpful in informing investigations into the brain.

Other useful continuations of this project include applying these methods to other parsers to see if any of them perform better than SyntaxNet. Discovering a parser with more correlations to the mechanisms of the brain would be extremely valuable information as that parser may be a better prediction of how the brain processes language. Alternatively, finding other parsers that do not perform as well as SyntaxNet would be helpful in the sense that researchers could begin to "rule out" those structures as plausible models for the brain's comprehension procession. Finally, it would also be interesting to compare these parsers to other kinds of brain data, namely those from functional neuroimaging. As we have seen, there is value in temporal comparisons between the brain and the parser, but with information about the spatial relationship between these two aspects, we might get a better look into exactly which processes we are examining.

Conclusion

There are several exciting conclusions we can draw from this project. The first is that there may in fact be some kind of correlation between the actions taken by a natural language processor and the EEG response elicited by people listening to natural language data. Specifically there is a relationship between the memory load of the parser and what appears to be the working memory load of the brain. Similarly there is a correlation between the occurrence of right arcs in the parser, and what may be the "end of sentence effect" described by Nelson & Dehaene (2017). These results support the idea that parsers like Parsey McParseface may be useful in finding a cognitively plausible model for the contents of the black box of language processing.

Another take-away from this project is that there is clearly a valuable relationship between computational linguistics and cognitive neuroscience. Both fields advancing to the point where they can inform each other's research. Knowing more about the brain can give AI developers direction when creating their agents. Similarly, the increasing accuracy of natural language processors is making them helpful in research into the brain's mechanisms of sentence comprehension. This relationship is one that is just beginning to be exploited in the research community, and one that we can only assume will build over time.

References

Abney, S. P., & Johnson, M. (n.d.). Memory Requirements and Local Ambiguities of Parsing Strategies.

Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., Ganchev, K., . . . Collins, M. (2016). Globally Normalized Transition-Based Neural Network. Retrieved from https://arxiv.org/pdf/1603.06042.pdf.

Brennan, J. (n.d.). Neuro-Computational Models of Natural Language. *US-French Research Proposal, Collaborative Research*. (Under Review)

Cowan, Nelson. (1996). Short-Term Memory, Working Memory, and Their Importance in Language Processing. Topics in Language Disorders. 17. 10.1097/00011363-199611000-00003.

Fletcher, P. C., & Henson, R. N. (2001). Frontal Lobes and Human Memory: Insights from Functional Neuroimaging. *Brain: A Journal of Neurology,124*(5), 849-881. Retrieved April 11, 2018, from https://doi.org/10.1093/brain/124.5.849.

Hale, J. T., Dyer, C., Kuncoro, A., & Brennan, J. R. (n.d.). Finding syntax in human encephalography with beam search. (Under review)

Kong, L., Alberti, C., Andor, D., Bogatyy, I., & Weiss, D. (2017). DRAGNN: A Transition-Based Framework for Dynamically Connected Neural Networks. Retrieved from https://arxiv.org/pdf/1703.04474.pdf.

Maris, E. & Oostenveld, R. (2007). Nonparametric statistical testing of EEG- and MEG-data. Journal of Neuroscience Methods, 164(1), 177–190.

Nelson, M. J., & Dehaene, S. (2017). Neurophysiological Dynamics of Phrase-Structure Building During Sentence Processing (J. T. Hale & C. Pallier, Eds.). *PNAS*.

Petrov, S. (2016, May 12). Announcing SyntaxNet: The World's Most Accurate Parser Goes Open Source. Retrieved from https://research.googleblog.com/2016/05/announcing-syntaxnet-worlds-most.html

Wu, N., & Bogatyy, I. (2017, November 13). SyntaxNet Tutorial. Retrieved from https://github.com/tensorflow/models/blob/master/research/syntaxnet/g3doc/syntaxnet-tutorial.md

Wu, N. (2017, September 21). Running DRAGNN Docker Image on the Cloud. Retrieved from https://github.com/tensorflow/models/blob/master/research/syntaxnet/g3doc/CLOUD.md

Author Note

This research was supported immensely by the advice and guidance of Dr. Jonathan Brennan, Assistant Professor, Department of Linguistics, University of Michigan. Without his help, this project would never have come to be. I must also thank his entire lab for their efforts in collecting the EEG data that made this project possible.

I am also thankful for the endless support of my friends and family, without whom I never would have completed this project. Their continual assurance and motivation are what got me through the day to day aspects of this project.

This thesis was conducted through an independent study course: LING 496, Section 040.

Correspondence concerning this project should be directed to Tyree Cowell, University of Michigan, Ann Arbor, MI, 48104

Contact: cowellty@umich.edu

Table 1

SyntaxNet Example Data

Focus Word	Left Arcs	Right Arcs	Stack Depth
Would	0	0	1
the	0	0	2
fall	0	0	3
never	1	0	3
come	0	0	4
to	3	0	2
an	0	0	3
end	1	3	0

This table shows what my data would look like for one sentence of Alice in Wonderland text. The three columns on the right show what the system looks like *after* the related word has been processed.

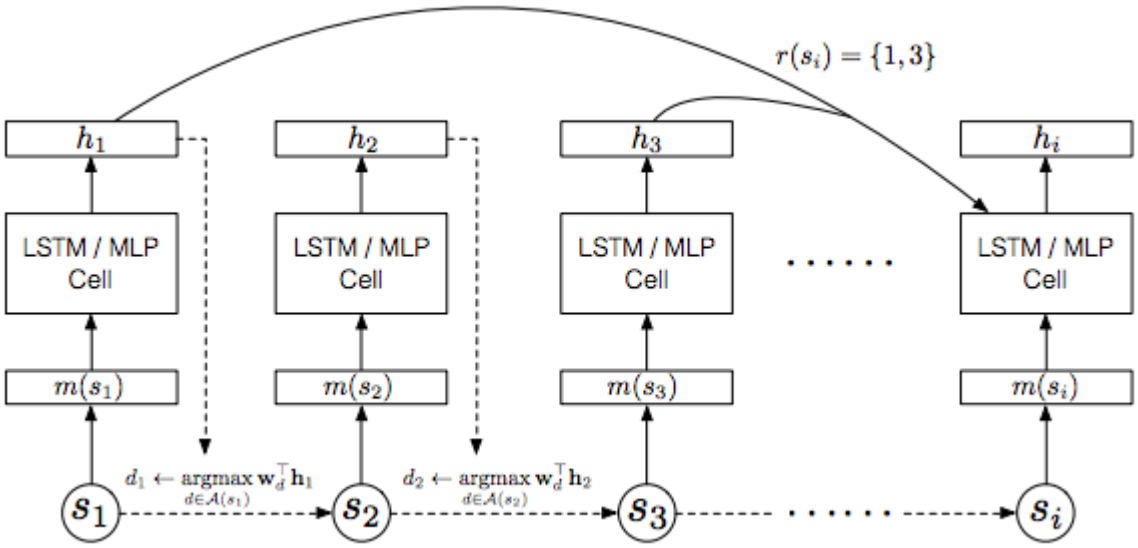

Figure 1: TBRU Framework. This figure shows the elements included in a TBRU system as well as how those elements relate to one another.

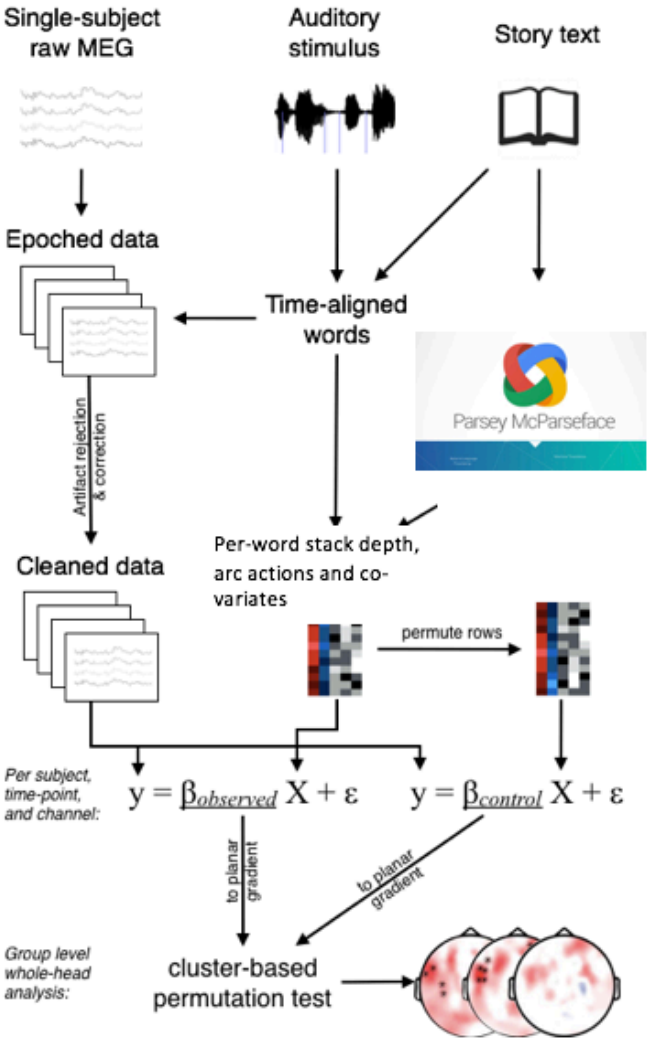

Figure 2: Data Combination Flow Chart. This figure shows how each data set (EEG and SyntaxNet) was prepared and combined using a simple linear regression. **Needs citation**

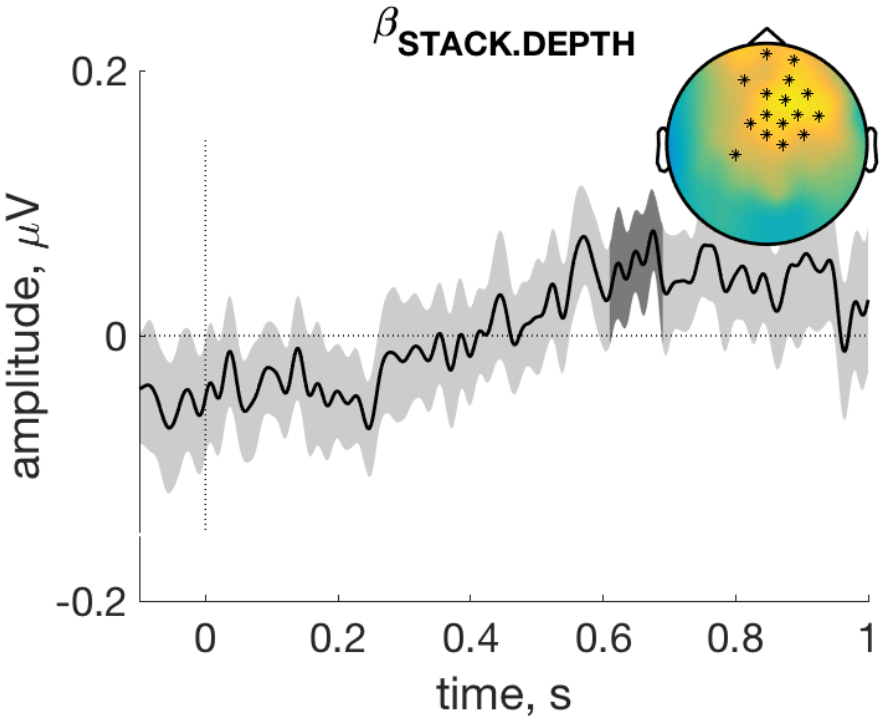

Figure 3: Stack Depth Regression Results. This graph shows the relationship between amplitudes measured by EEG and the changing stack depth of Parsey McParseface. The x-axis is time, starting at word onset. The y-axis represents the unit change in amplitude for every unit change in stack depth. The section that is shaded darker represents a cluster of Beta values that significantly differ from zero. The circle on the right is an aerial view of a human head. The asterisks portray the electrodes that significantly differed, reflecting the average amplitude across the shaded time interval. Warmer colors represent more positive amplitudes, while cooler colors represent negative ones.

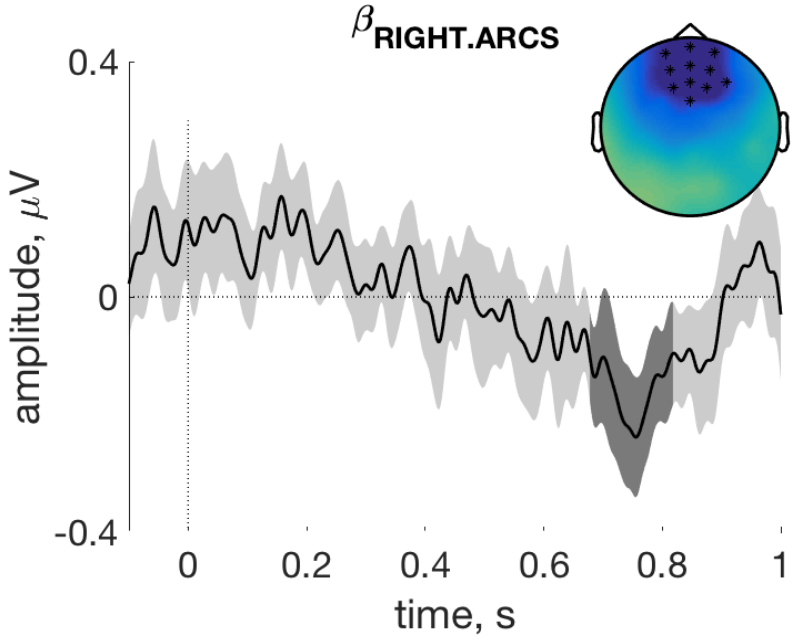

Figure 3: Right Arc Regression Results.
This graph shows the relationship between amplitudes measured by EEG and the number of right arcs made by the parser. The x-axis is time, starting at word onset. The y-axis represents the unit change in amplitude for every unit change in stack depth. The section that is shaded darker represents a cluster of Beta values that significantly differ from zero. The circle on the right is an aerial view of a human head. The asterisks portray the electrodes that significantly differed, reflecting the average amplitude across the shaded time interval. Warmer colors represent more positive amplitudes, while cooler colors represent negative ones.

Appendix A: Alice in Wonderland Text

*Note: The following sentences are those used as stimuli for both Parsey McParseface and the EEG participants in **citation**, they have been processed so that every letter is lower-cased, each sentence is tokenized, and the entire text is separated into discrete sentences. Text taken from Carroll, L., & Blum, A. A. (1968). Alice in Wonderland. New York: Gilberton.*

alice was beginning to get very tired of sitting by her sister on the bank and of having nothing to do

once or twice she had peeped into the book her sister was reading but it had no pictures or conversations in it and what 's the use of a book thought alice without pictures or conversation

so she was considering in her own mind as well as she could for the hot day made her feel very sleepy and stupid whether the pleasure of making a daisy chain would be worth the trouble of getting up and picking the daisies when suddenly a white rabbit with pink eyes ran close by her

there was nothing so very remarkable in that nor did alice think it so very much out of the way to hear the rabbit say to itself oh dear

oh dear

i shall be late

when she thought it over afterwards it occurred to her that she ought to have wondered at this but at the time it all seemed quiet natural

but when the rabbit actually took a watch out of its waistcoat pocket and looked at it and then hurried on alice started to her feet for it flashed across her mind that she 'd never before seen a rabbit with either a waistcoat pocket or a watch to take out of it

and burning with curiosity she ran across the field after it and fortunately was just in time to see it pop down a large rabbit hole under the hedge

in another moment down went alice after it never once considering how in the world she was to get out again

the rabbit hole went straight on like a tunnel for some way and then dipped suddenly down

so suddenly that alice had not a moment to think about stopping herself before she found herself falling down a very deep well

either the well was very deep or she fell very slowly for she had plenty of time as she went down to look about her and to wonder what was going to happen next

first she tried to look down and make out what she was coming to but it was too dark to see

anything

then she looked at the sides of the well and noticed that they were filled with cupboards and bookshelves here and there

she saw maps and pictures hung upon pegs

she took down a jar from one of the shelves as she passed

it was labeled orange marmalade but to her great disappointment it was empty

she did not like to drop the jar for fear of killing somebody so she managed to put it in to one of the cupboards as she fell past it

well thought alice to herself after such a fall as this i shall think nothing of tumbling down stairs

how brave they 'll all think me at home huh why i would n't say anything about it even if i fell off the top of the house which was very likely true down down down

would the fall never come to an end

i wonder how many miles i 've fallen by this time she said aloud

i must be getting somewhere near the center of the earth

let me see that would be four thousand miles down i think

for you see alice had learned several things of this sort in her lessons in the schoolroom and though this was not a very good opportunity for showing off her knowledge as there was no one to listen to her still it was good practice to say it over

yes that 's about the right distance but then i wonder what latitude or longitude i 've got to

alice had no idea what latitude was or longitude either but thought they were nice grand words to say

presently she began again i wonder if i shall fall right through the earth

how funny it 'll seem to come out among the people that walk with their heads downward

the antipathies i think

she was rather glad there was no one listening this time as it did n't sound at all the right word

but i shall have to ask them what the name of the country is you know please ma'am is this new zealand or australia

and she tried to curtsey as she spoke fancy curtseying as you 're falling through the air

do you think you could manage it

and what an ignorant little girl she 'll think me for asking

no it 'll never do to ask

perhaps i shall see it written up somewhere

down down down

there was nothing else to do so alice soon began talking again

dinah 'll miss me very much tonight i should think

dinah was the cat

i hope they 'll remember her saucer of milk at tea time

dinah my dear i wish you were down here with me

there 're no mice in the air i 'm afraid

but you might catch a bat and that 's very like a mouse you know

but do cats eat bats i wonder

and here alice began to get rather sleepy and went on saying to herself in a dreamy sort of way

do cats eat bats

do cats eat bats

and sometimes do bats eat cats

for you see as she could n't answer either question it did n't much matter which way she put it

she felt that she was dozing off and had just begun to dream that she was walking hand in hand with dinah and saying to her very earnestly

now dinah tell me the truth did you ever eat a bat

when suddenly thump thump down she came upon a heap of sticks and dry leaves and the fall was over

alice was not a bit hurt and she jumped up on her feet in a moment

she looked up but it was all dark overhead

before her was another long passage and the white rabbit was still in sight hurrying down it

there was not a moment to be lost

away went alice like the wind and was just in time to hear it say as it turned a corner

oh my ears and whiskers how late it 's getting

she was close behind it when she turned the corner but the rabbit was no longer to be seen

she found herself in a long low hall which was lit up by a row of lamps hanging from the roof

there were doors all around the hall but they were all locked

and when alice had been all the way down one side and up the other trying every door she walked sadly down the middle wondering how she was ever to get out again

suddenly she came upon a little three legged table all made of solid glass

there was nothing on it except a tiny golden key and alice 's first thought was that it might belong to one of the doors of the hall

but alas either the locks were too large or the key was too small but at any rate it would not open any of them

however on the second time round she came upon a low curtain she had not noticed before and behind it was a little door about fifteen inches high

she tried the little golden key in the lock and to her great delight it fitted

alice opened the door and found it led into a small passage not much larger than a rat hole

she knelt down and looked along the passage into the loveliest garden you ever saw

how she longed to get out of that dark hall and wander about among those beds of bright flowers and those cool fountains

but she could not even get her head through the doorway

and even if my head would go through thought poor alice it would be of very little use without my shoulders

oh how i wish i could shut up like a telescope

i think i could if i only know how to begin

for you see so many out of the way things had happened lately that alice had begun to think that very few things indeed were really impossible

there seemed to be no use in waiting by the little door so she went back to the table half hoping she might find another key on it or at any rate a book of rules for shutting people up like telescopes

this time she found a little bottle on it which certainly was not here before said alice

and round the neck of the bottle was a paper label with the words drink me ' beautifully printed on it in large letters

it was all very well to say drink me but the wise little alice was not going to do that in a hurry

no i 'll look first she said and see whether it 's marked poison or not

for she had read several nice little histories about children who got burnt and eaten up by wild beasts and other unpleasant things all because they would not remember the simple rules their friends had taught them

such as that a red hot poker will burn you if you hold it too long

and that if you cut your finger very deeply with a knife it usually bleeds

and she had never forgotten that if you drink much from a bottle marked poison it 's almost certain to disagree with you sooner or later

however this bottle was not marked poison so alice ventured to taste it and finding it very nice

it had in fact a sort of mixed flavor of cherry tart custard pineapple roast turkey toffee and hot buttered toast

she very soon finished it off

what a curious feeling said alice i must be shutting up like a telescope

and so it was indeed

she was now only ten inches high and her face brightened up at the thought that she was now the right size for going through the little door into that lovely garden

first however she waited for a few minutes to see if she was going to shrink any further

she felt a little nervous about this for it might end you know said alice to herself in my going out all together like a candle

i wonder what i should be like then and she tried to fancy what the flame of a candle is like after the candle is blown out for she could not remember having ever seen such a thing

after a while finding that nothing more happened she decided on going in to the garden at once

but alas for poor alice

when she got to the door she found she 'd forgotten the little golden key

and when she went back to the table for it she found she could not possibly reach it

she could see it quite plainly through the glass and she tried her best to climb up one of the legs of the table but it was too slippery

and when she had tired herself out with trying the poor little thing sat down and cried

come there 's no use crying like that said alice to herself rather sharply

i advise you to leave off this minute

she generally gave herself very good advice though she very seldom followed it

and sometimes she scolded herself so severely as to bring tears into her eyes

she once remembered trying to box her own ears for having cheated herself in a game of croquet she was playing against herself

for this curious child was fond of pretending to be two people

but it 's no use now thought poor alice to pretend to be two people

why there 's hardly enough of me to make one respectable person

soon her eye fell on a little glass box that was lying under the table

she opened it and found in it a very small cake on which the words eat me were beautifully marked in currants

well i 'll eat it said alice and if it makes me grow larger i can reach the key and if it makes me grow smaller i can creep under the door so either way i 'll get into the garden and i do n't care which happens

she ate a little bit and said anxiously to herself which way which way holding her hand on the top of her head to feel which way it was growing

and she was quite surprised to find that she remained the same size

to be sure this generally happens when one eats cake

Appendix B: Code for Data Extraction

```
//  main.cpp
//  dragnn_parser
//
//  Created by Tyree Cowell on 12/28/17.
//  Copyright (c) 2017 Tyree Cowell. All rights reserved.
//

#include <iostream>
#include <vector>
#include <string>
#include <fstream>
#include <cstdlib>

using namespace std;

//struct that stores how many arcs for a shift
//and the stack depth after that shift
struct Shift {
   int LEFT_ARCs = 0;
   int RIGHT_ARCs = 0;
   int stack_depth = 0;
};

void update_shift(vector<Shift> &final, vector<int> &decisions, int depth) {

   //initialize a shift object
   Shift update;
   //initilize that shift's stack_depth to depth
   update.stack_depth = depth;

   //loop through all the decisions for this shift
   for (int i = 0; i < decisions.size(); ++i) {

      //if you've reached a shift, that should be the end of the vector
      if (decisions[i] == 1) { break; }

      //if you reach a left arc, increase the number of left arcs for this shift
      else if (decisions[i] == 2) { ++update.LEFT_ARCs; }

      //do the same if you reach a right arc
      else if (decisions[i] == 3){ ++update.RIGHT_ARCs; }
   }

   //add the shift data to the final table of shifts
```

```
    final.push_back(update);
    return;
}

void print_parser(vector<Shift> &final, vector<string> text) {

    //print header
    cout << "FOCUS  LEFT_ARCS  RIGHT_ARCS  STACK DEPTH" << endl;

    //loop through the entire sentence in text
    for (int i = 0; i < text.size(); ++i) {
        //print out the focus word
        cout << text[i] << "  ";
        //print out the shift data for that word
        cout << final[i].LEFT_ARCs << "    " << final[i].RIGHT_ARCs
        << "    " << final[i].stack_depth;

        cout << endl;
    }

    return;
}

string get_filename(int sentence_num) {

    char c = '0' + sentence_num;
    string filename = "output_";

    if(sentence_num < 10) {
        filename += c;
    }

    else if(sentence_num > 9 && sentence_num < 100) {
        int tens = sentence_num / 10;
        int ones = sentence_num % 10;
        filename += ('0' + tens);
        filename += ('0' + ones);
    }

    else if (sentence_num > 99){
        filename += '1';
        sentence_num -= 100;

        int tens = sentence_num / 10;
        int ones = sentence_num % 10;
        filename += ('0' + tens);
```

```
        filename += ('0' + ones);

        sentence_num += 100;
    }

    filename += ".txt";

    return filename;
}

vector<Shift> make_table(int sentence_num) {

    //initialize final table of shifts
    vector<Shift> final;
    ifstream in;

    //create filename from sentence_num
    filename = get_filename;

    //open output file
    in.open(filename);

    //initialize stack depth at 0
    int stack_depth = 0;
    int decision;
    in >> decision;

    //read in all decisions from output file
    while (decision > 0) {

        //initialize vector of decisions for each word
        vector<int> decisions;

        //read in decisions until you find a shift (1)
        //will not enter this loop if the first decision was a shift
        while (decision > 1) {

            //the deicison is an arc, decrease the stack depth
            --stack_depth;

            //add the decision to the vector of decisions
            decisions.push_back(decision);

            //read in the next decision
            in >> decision;
```

```
        }

        //add the shift to the vector of decisions
        decisions.push_back(decision);

        //since you must have reached a shift to get to this point
        //increase the stack depth
        //unless you are at the end of the file
        if (decision != 0) {
            ++stack_depth;
        }

        //create a shift entry, and add it to the final table
        update_shift(final, decisions, stack_depth);

        in >> decision;
    }

    //return the final table
    return final;

}

//splits a string (text) into a vector of words
vector<string> split(string text) {

    istringstream iss(text);
    vector<string> words((istream_iterator<string>(iss)), istream_iterator<string>());

    return words;
}

int main(int argc, const char * argv[]) {

    //initialize vector of all sentences
    vector<string> alice_text;
    alice_text.resize(113);

    //open alice text file
    ifstream in;
    in.open("test.txt");

    int i = 0;
    string get;

    //initialze final vector of all shifts
```

```
    vector<vector<Shift>> all;

    //make a shift table for each sentence
    //add each table to all
    while (i < 117) {
        getline(in, get);
        alice_text[i] = get;
        vector<Shift> next = make_table(i);
        all.push_back(next);
        ++i;
    }

    in.close();

    //ask user which table they want to output
    cout << "which sentence number would you like? (enter -1 to quit)" << endl;
    int sentence_num = 0;
    cin >> sentence_num;

    //loops until the user quits by entering -1
    while(sentence_num != -1) {

        //break chosen sentence into words for easier output
        vector<string> sentence = split(alice_text[sentence_num]);

        //print table
        print_parser(all[sentence_num], sentence);

        //ask for a new sentence number
        cout << "which sentence number would you like? << endl; << (enter -1 to quit)" << endl;
        cin >> sentence_num;

    return 0;
}
```