An Interactive Activation Model of Word Recognition

Andy Valenti

October 19, 2017

1 Program Notes

1.1 Program Files

The Interactive Activation program consists of two files:

IA.py main script which implements the IA model's activation function and other characteristics, provides a simple, menu-based interface, implements a few ways to graphically display the model's letter and word activations, and initiates the model's script processor.

IA_pools.py contains the model's data structures which implement the letter and word levels of the model

1.2 Program Requirements

The following items must be installed prior to running the program:

- python 2.7
- matplotlib.pyplot

Alternatively, the *anaconda* distribution can be downloaded and installed which 'includes over 100 of the most popular Python, R and Scala packages for data science' (including matplotlib). For more information, see (*Anaconda Python Distribution*, 2017)

1.3 Running the Program

type:

```
python IA.py
```

2 Basic Concepts of the Interactive Activation Model

Our model is based on the J. McClelland and Rumelhart (1981) classic interactive activation (IA) model of letter perception as extended to the bilingual case by Dijkstra and Heuven (2002); it is commonly known as the BIA model. The BIA model was further enhanced by Dijkstra and VanHeuven to incorporate the concept of inhibitory control (Green, 1998) which proposes that the cost of switching languages is due to the inhibition of the non-active language's lexicon by the active language. This model, enhanced with other concepts as well, is commonly referred to as the BIA Plus model. Our model most closely resembles BIA Plus.

In the IA model, perception is suggested to arise across a set of interacting levels, organized as a hierarchy of features, letters, and words. The theory suggests information flowing 'top-down' combines with information flowing 'bottom-up' to provide a set of constraints out which perception arises (J. McClelland & Rumelhart, 1981). Information flows through a a spreading activation process so that information at one level spreads to neighboring levels above and below. While it is likely that there are many levels contributing to perception, J. McClelland and Rumelhart (1981) suggest that examining the interaction between the word and letter levels is sufficient to explain many word perception effects.

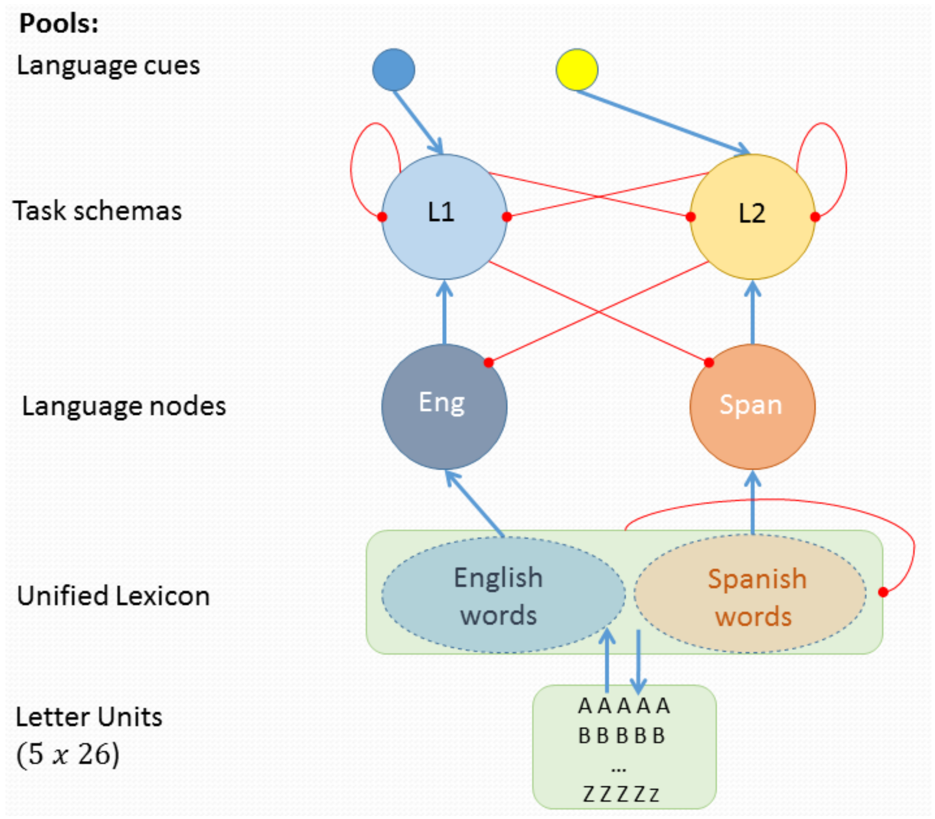


Figure 1: Bilingual Interactive Activation model processing levels. Processing levels are connected according to the type of connectors shown. Connectors (red) terminating with circles are the inhibitory connections and those with arrow heads (blue) are the excitatory connections. applied in the direction of terminators.

2.1 Basic IA Model Assumptions

The following information is adapted from (J. L. McClelland & Rumelhart, 1989).

Perception occurs in a multilevel processing system: The model has separate representations for visual features (i.e., one of 14 line segments which compose a letter, like in an LED segment display), letters, words and large wholes e.g., sentences. The IA portion of this model has two levels: a letter and a word level.

Deeper levels of processing are accessed through intermediate levels: For example, the letter level resides in between the visual feature and the word levels. This is because J. L. McClelland and Rumelhart propose that words appear to be defined not because of their visual configuration, but because of the sequence of letters they contain (e.g., READ read *read* are all identifiable as words and the letters in them are perceived more accurately than the same letters in an unrelated context, see Section 8.1).

Processing is interactive: J. L. McClelland and Rumelhart mean that processing involves considering both bottom-up and top-down information in parallel (see Section 2). The role of word context in letter perception is an example in support of this assumption.

Information flow is continuous: As opposed to information processing occurring through a sequence of steps, which was the prevailing view prior to development of the IA model. J. L. McClelland and Rumelhart assumed continuous processing is required to account for word contextual influence in word perception. They reason that if word processing were to influence letter processing, then the letter level needs to make available its information available to the word level prior to completing letter level processing.

3 Model Dynamics

The processing dynamics in our implementation of the Interactive Activation model are based on a class of models known as interactive activation and competition (IAC) models. Refer to J. L. McClelland (2015, sect. 2.2, The IAC Model) for a thorough description; we present a summary of the relevant concepts in the following section and contrast where our model differs. The processing levels of the model is shown Figure 1.

Architecture: The IAC model consists of sets of units divided into pools; in each pool all of the units inhibit each other. Our IA model follows this design. Between pools, units may have excitatory or inhibitory connections. In our IA model some of these connections are *not* bi-directional while others are not (whereas they usually all are in IAC).

Cycles: The activations of the units in an IAC network evolve gradually over time in a continuous fashion. However, when simulating this model computationally this mathematical ideal is approximated by breaking time into a sequence of discrete steps called cycles. At the start of every cycle, the activation value of every unit is the value that was computed at the end of the preceding cycle. Here is how the network computes the values:

1. Compute the input values to each unit
2. Compute the activation of the units

This two-step procedure ensures that nothing is done with the new activation of any of the units until all have been updated (i.e., the update is synchronous).

Activation: As per J. L. McClelland (2015), the change in activation of the units in an IAC network is a function that takes as input the current activation of the unit and the net input to the unit from other units or from outside the network. The net input to a given unit (e.g., unit i) is the sum of the influences of all of the other units in the network plus any external input from outside the network. The influence of another unit (e.g., unit j) is the product of that unit's output, $output_j$, multiplied by the weight of the connection to unit i from unit j. Thus the net input to unit i is given by Equation 1:

$$net_i = \sum_j W_{ij} output_j + extinput_i \tag{1}$$

In the IAC model, $output_j = [a_j]^+$. Here, a_j refers to the activation of unit j, and the expression $[a_j]^+$ has the value a_j for all $a_j > 0$; else its value is 0. This type of activation function is sometimes called a rectified linear neuron or a linear threshold neuron.

The value of index j is the range over all of the units that are connected to unit i and the weights w can be positive or negative, for excitatory or inhibitory connections, respectively.

Once the net input to a unit has been computed, the resulting change in the activation of the unit is as follows:

If ($net_i > 0$):

$$\Delta a_i = (max - a_i)net_i - decay(a_i - rest) \tag{2}$$

else:

$$\Delta a_i = (a_i - min)net_i - decay(a_i - rest) \tag{3}$$

Parameters: The IAC model has several parameters which can be modified by the user. They are discussed in Section 4.13.

3.1 Model Pools

As described in the preceding section, an IAC model consists of a set of processing units organized into several pools. Excitatory connections exist between units in different pools and inhibitory connections between units within the same pool. Processing is considered to be interactive because each pool both influences and is influenced by processing in other pools. The inhibitory connections within a pool implements competition among the units so that the units in the pool that receive the strongest activation tend to drive down the activation of the other units.

Consistent with the IAC architecture, our Interactive Activation (IA) model contains five pools of:

1. Letters: When a word is presented as external input to the model, the associated letters are activated. To keep the set of words manageable, the input must be less than or equal to five letters; as a result there are five sets of 26 letters, A - Z, each set corresponding to a position in a word.

   All letters in the i^{th} set have excitatory connections to words that contain them in the i^{th} position and inhibitory connections to all words with a different letter in that position. Thus the unit for T in the first position excites $TAKE$ but not $CART$, $STOP$, etc.

2. Words: Consistent with the language non-selective access principle of the BIA model, which assumes words in both languages are potential candidates for selection and actively compete with one another, words from both languages are combined in the same pool.

   Each word projects an excitatory connection from its i^{th} letter to the corresponding letter in the i^{th} set in the letter pool. Words in the pool have inhibitory connections between each other. For example, there is an excitatory connection from $TAKE$ to T in the first position, to A in the second, etc.

   Within the word pool, there are inhibitory connections used to suppress mutually exclusive interpretations of the input.

3. Languages: English and Spanish. Consistent with the BIA Plus model, the language nodes serve four functions:

   (a) Language tags (labels) indicating the language to which the word belongs

   (b) Activation accumulators which could account for between-language priming effects

   (c) Functional mechanism which modulates relative language activation, i.e., language filter rather than a switch

   (d) Serves as a collection point for context pre-activation from outside the word recognition system

4. Cues: English (L1) and Spanish (L2). The cues indicate the language that is the focus of the participant's attention. For example, we modeled the lexical decision task described in (von Studnitz & Green, 1997) in which the presence of an external cue (i.e., the color of the background on which the letter string is presented) informs participants of the required language for decision. This script for this experiment is shown in Section 5.9.

5. Schemas: Two lexical decision schemas, L1 (English) and L2 (Spanish) are hypothesized by Green (1998) to be established in order to execute a person's intention to perform a specific language task such as Lexical Decision (LD). In modeling an LD task, for example, the schema relates an output of the lexical system (e.g. L1 tag present) to an LD response.

3.2 Conducting an Experiment

The model is designed to simulate trials from psychological experiments. A trial is a sequence of stimuli presented to the model for processing. We will adopt the terminology used by J. L. McClelland and Rumelhart and henceforth called the stimulus data a 'field'. It is possible to present a blank field to the model (Section 4.5) When the model is started, the trial number is set to 1 and the cycle number is set to 0; the network activations are set to their default baseline values. The first field is presented to the model at the beginning of cycle 1, followed by the second field, etc. When the model is reset, the trial number is increased by 1; the cycle number is reset to 0 and the network activations to their baseline levels.

The processing of a sequence of fields continues for the desired number of cycles after which there may be a lexical decision task, i.e., is the presented input an English word or is it not an English word? or a forced-choice test, i.e., given a cue to a position in a word and a choice between two letters, which was the letter that actually occurred in the position pointed to be the cue?

3.3 Input to the Model

The inputs to the model, which are the external fields presented during processing trials, are the letter units which should be turned. The fields presented may be deterministic (i.e., all letters specified may be turned on or off) or it may be stochastic (i.e., specified letters may be detected with some probability). The probability may vary from field to field to simulate the possibility that the input is more or less decipherable from case to case. In the current version of the module, the external fields are deterministic.

4 Interactive Activation Model Functions

4.1 User interface (UI)

The program offers a simple, menu-driver user interface as shown in the following section. The interface is divided into three portions: Main Menu, Console Message, and the Command Line.

4.2 Conventions used in this guide

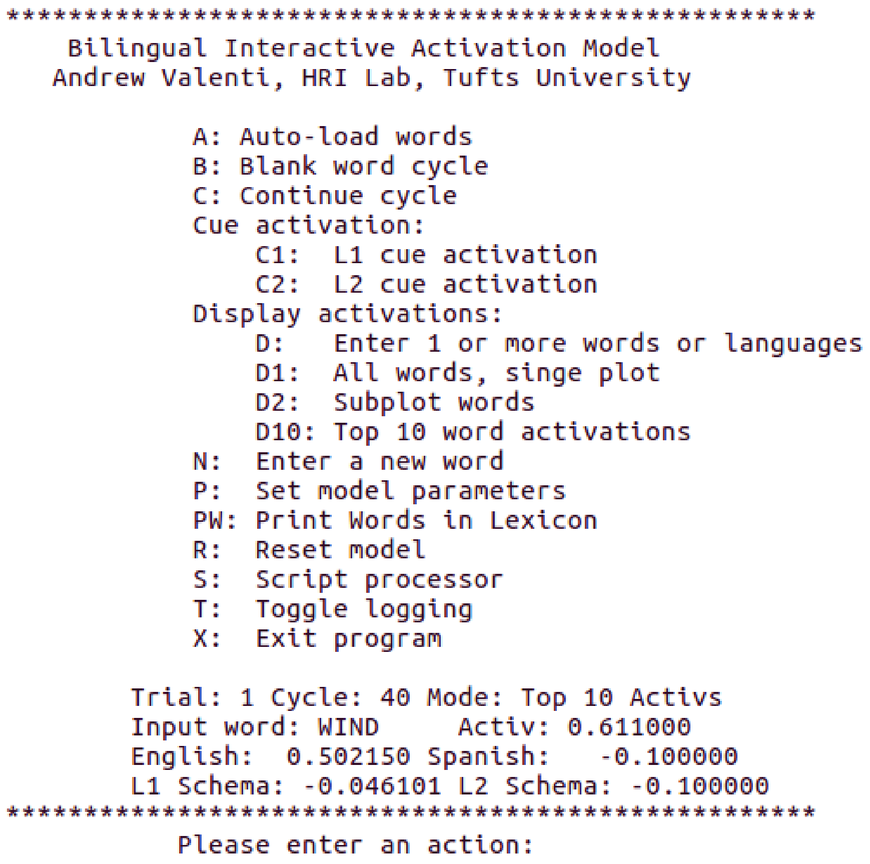
Italics
Indicates new terms, URLs, email addresses, filenames, etc.

[terms]
Indicates terms that are optional

Command
Indicates the 1 or 2 characters of description which can be used to invoke a command, e.g., **C**ontinue cycle
All commands can be provided in either upper or lower case.

4.3 Main Menu

The Main Menu provides a character-based interface to the program's functions. Each line of the menu offers a concise description of the function, preceded by ts one or two character execution command. Commands are entered in the Command Line, the portion of the UI which is after the menu and the console message areas. An invalid command will invoke the error message: *Unrecognized action. Try again.*



```
****************************************************
    Bilingual Interactive Activation Model
   Andrew Valenti, HRI Lab, Tufts University

            A: Auto-load words
            B: Blank word cycle
            C: Continue cycle
            Cue activation:
                C1:  L1 cue activation
                C2:  L2 cue activation
            Display activations:
                D:   Enter 1 or more words or languages
                D1:  All words, singe plot
                D2:  Subplot words
                D10: Top 10 word activations
            N:  Enter a new word
            P:  Set model parameters
            PW: Print Words in Lexicon
            R:  Reset model
            S:  Script processor
            T:  Toggle logging
            X:  Exit program

        Trial: 1 Cycle: 40 Mode: Top 10 Activs
        Input word: WIND     Activ: 0.611000
        English:  0.502150 Spanish:   -0.100000
        L1 Schema: -0.046101 L2 Schema: -0.100000
****************************************************
           Please enter an action:
```

Figure 2: Interactive Activation Model Main Menu.

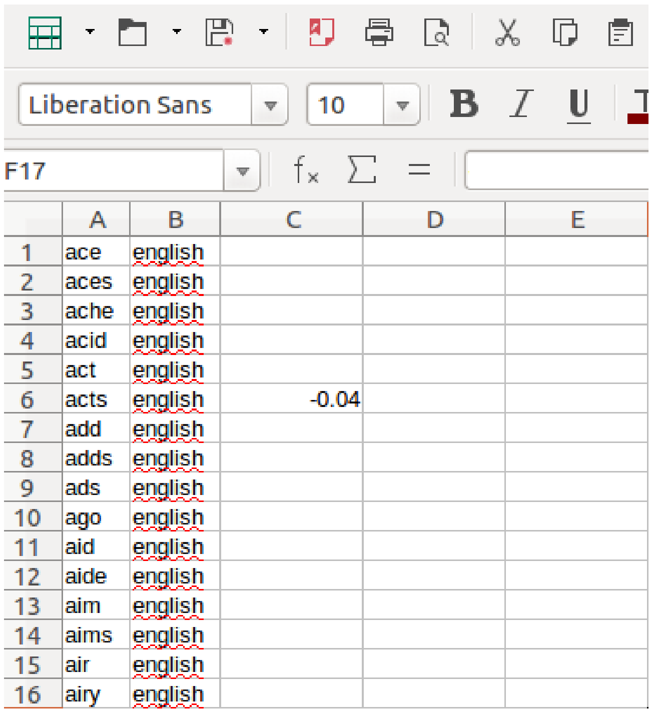
4.4 Auto-load words

command: A

purpose: Reads new word stimuli from a file with the extension *.csv* and updates the *lets, words, and lang* pools.

file format: one entry per line in the CSV format: *word,language,[resting activation]*
The language is from the set {english, spanish} and is required. The optional resting activation must be a floating point number. If no number is supplied, the current value of the default resting activation, *rest* (-0.1), is used. This default may be changed as described in Model Parameters, Section 4.13.

It is recommended that the file be created in a spreadsheet program such as *Microsoft Excel* or *LibreOffice Calc* and saved as a CSV text file. The advantage of doing so is that each entry can be stored in a separate spreadsheet cell and the program will ensure that the file is formatted with the proper value and line delimiters when it is saved (see Figure 3).



	A	B	C	D	E
1	ace	english			
2	aces	english			
3	ache	english			
4	acid	english			
5	act	english			
6	acts	english	-0.04		
7	add	english			
8	adds	english			
9	ads	english			
10	ago	english			
11	aid	english			
12	aide	english			
13	aim	english			
14	aims	english			
15	air	english			
16	airy	english			

Figure 3: Example of part of a spreadsheet for auto-loading words

notes: $3 \leq$ wordlength ≤ 5. Words loaded using this command are not saved between executions of the program and must be reloaded.

error messages:

Stimuli file does not exist *filename*: The supplied *filename* cannot be found.

Stimuli improperly formatted: At least the word and its language need to be specified.

Unknown language attribute *language*: The specified *language* is invalid.

Stimulus 3 <= **word** <= 5 **lets:** A word in the stimulus file is of improper length; it must be at least 3 and at most 5 letters in length.

One or more resting values defaulted: This is a warning that at least one word in the stimulus file does not have and associated resting level; the value of the default resting parameter, *rest* will be used.

4.5 Blank word cycle

command: B

purpose: Clear the external input (ExtInput) to the letters and the cues; cycles the network.

4.6 Continue cycle

command: C

purpose: Performs an update cycle so long as a word has been inputted; else the user is informed via the program's console message at bottom of menu.

error messages:

Unable to cycle until a word is entered: At least one word must have been entered into the model before this command can be invoked.

4.7 Cue activation

command: C1 (L1, English), C2 (L2, Spanish)

purpose: Activates a language cue by setting the desired cue's ext input to 1.

notes: The language cue represents the language the participant should be attending to: in this model, either English (L1) or Spanish (L2).

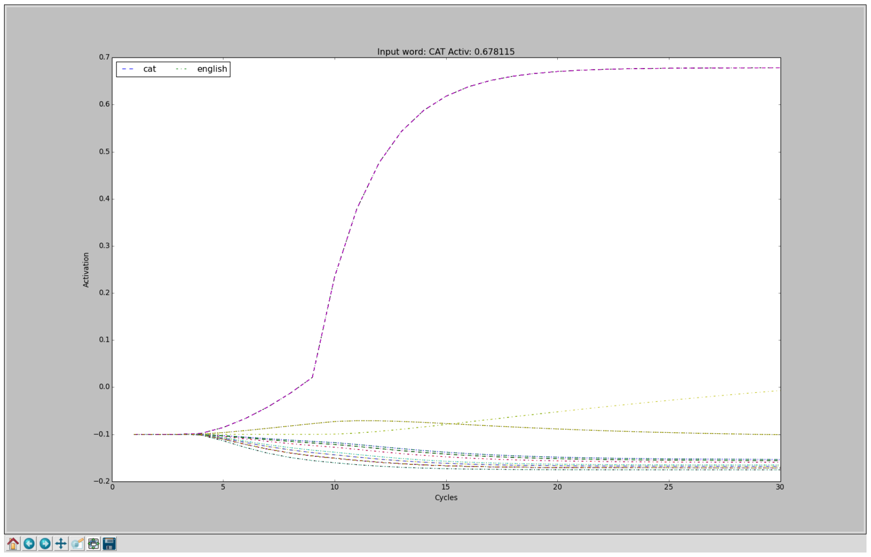


Figure 4: **D**isplay 1 or more words or languages

4.8 Display 1 or more words or languages (Figure 4)

command: D

prompt: Enter 1 or more items separated by a comma:

cat,gato,english

purpose: Allows comparison of word and language activations on a single plot. At the top of chart, the word that was last input via the **N**ew word command is displayed.

error messages:

No data available until a cycle is run: At least one word must have been entered into the model and the model must have been run for at least one cycle

Item not found in pool: The requested item to be charted cannot be found in any of the pools.

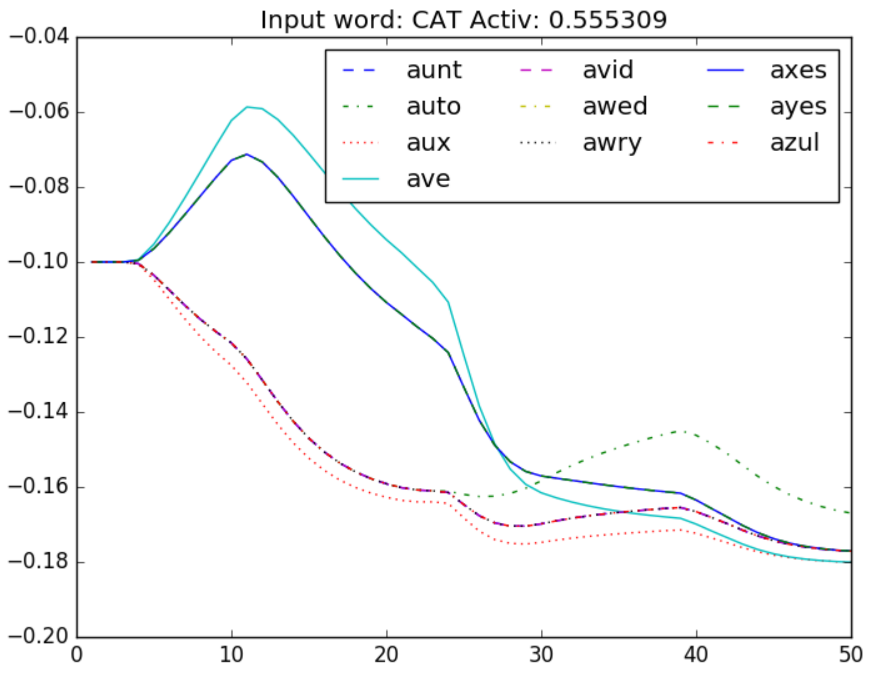


Figure 5: **D1** 10 words at-a-time, single plot.

4.9 Ten words at-a-time, single plot (Figure 5)

command: D1

prompt: Enter [N]ext, [P]rev, [Q]uit.

- **N** to scroll to the next 10 words, in alphabetical order. Will repeatedly display last set if there are no more words.
- **P** to scroll to the previous 10 words. Will repeatedly display first set if there are no more words.
- **Q** to quit the D1 display mode and close its chart.

purpose: Allows comparison of the activation of 10 words at-a-time in a single plot. At the top of chart, the activation of the word that was last input via the **N**ew word command is displayed. This is also a handy way to scroll through all words in the model's lexicon.

error messages:

No data available until a cycle is run: At least one word must have been entered into the model and the model must have been run for at least one cycle

? not valid action: A letter other than **N, P, Q** was entered at the prompt.

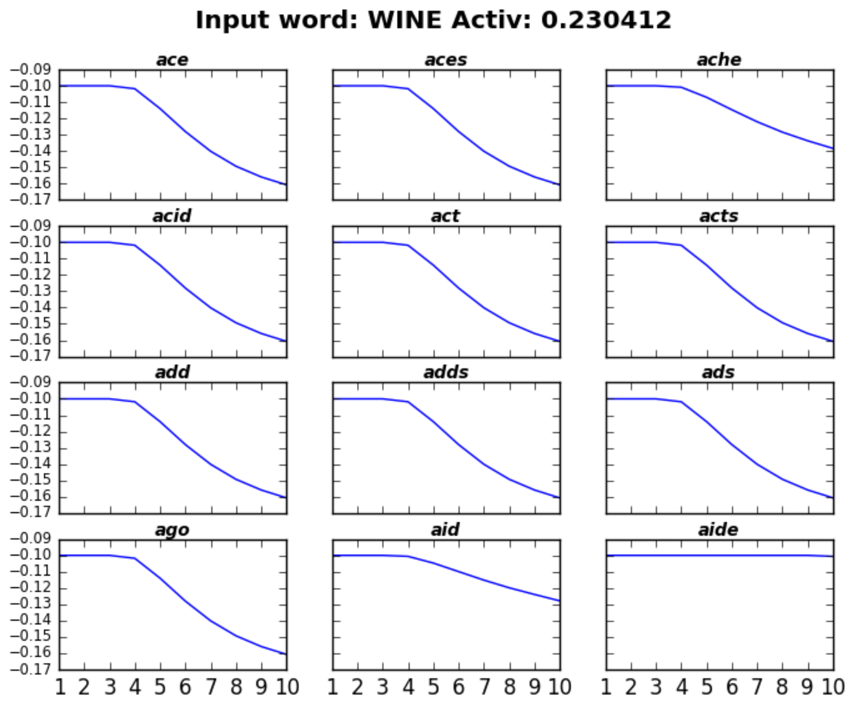


Figure 6: **D2** 12 words at-a-time, subplots.

4.10 Twelve words at-a-time, subplots

command: D2

prompt: Enter [N]ext, [P]rev, [Q]uit.

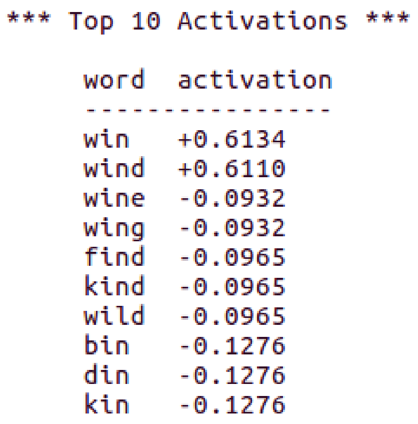
- **N** to scroll to the next 12 words (or as many exist), in alphabetical order. Will repeatedly display last set if there are no more words.
- **P** to scroll to the previous 12 words. Will repeatedly display first set if there are no more words.
- **Q** to quit the D1 display mode and close its chart.

purpose: Allows comparison of the activation of up to 12 words at-a-time, as subplots. At the top of chart, the activation of the word that was last input via the **N**ew word command is displayed.

error messages:

No data available until a cycle is run: At least one word must have been entered into the model and the model must have been run for at least one cycle

? not valid action: A letter other than **N, P, Q** was entered at the prompt.



```
*** Top 10 Activations ***

     word  activation
     ----------------
     win   +0.6134
     wind  +0.6110
     wine  -0.0932
     wing  -0.0932
     find  -0.0965
     kind  -0.0965
     wild  -0.0965
     bin   -0.1276
     din   -0.1276
     kin   -0.1276
```

Figure 7: **D10** Top 10 word activations.

4.11 Top 10 word activations

command: D10

purpose: to display the top 10 largest word activations in descending order for the last time step.

error messages:

No data available until a cycle is run: At least one word must have been entered into the model before this command can be invoked.

4.12 Enter a new word

command: N

purpose: Allows the user to enter a single word regardless of whether it is already in the lexicon as input to the model.

note: A list of words in the lexicon can be viewed by the Print All Words command i.e., **PW** (see Section 4.14).

error messages:

Word truncated to *newword*: A word longer than 5 characters was entered. The model used the first 5 characters to form *newword*

4.13 Set model parameters

command: P

purpose: Allows the user to change the default model parameters. Once reset, the new values will be redisplayed in the console message area automatically. To see the current value of the parameters, just hit **enter** immediately after. These parameters are *global* in scope i.e., they affect all pools in the model identically.

prompt: Enter params separated by a comma.
ncycles=20,alpha=0.2

note: The model's parameters are as given in *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises*, see (J. L. McClelland, 2015, sect. 2.2.4 Parameters). These are as follows:

max (1.0) The maximum activation parameter.

min (−0.2) The minimum activation parameter.

rest (−0.1) The resting activation level to which activations tend to settle in the absence of external input. Changing this parameter *only* sets a new default resting level during the construction of word units when auto-loading in the absence of any explicit resting level values in the auto-load file (see Section 4.4). It also does not affect the resting levels of the default word units or of any other pools.

decay (0.1) The decay rate parameter, which determines the strength of the tendency to return to resting level.

estr (0.4) This parameter stands for the strength of external input (i.e., input to units from outside the network). It scales the influence of external signals relative to internally generated inputs to units.

alpha (0.1) This parameter scales the strength of the excitatory input to units from other units in the network.

gamma (0.1) This parameter scales the strength of the inhibitory input to units from other units in the network.

error messages:

Invalid parameter keyword: Keyword must be one of the names of the parameters as given above.

4.14 Print Words

command: PW

purpose: Prints to the terminal all words in the lexicon, in alphabetical order.

4.15 Reset model

command: R

purpose: Removes input activation to the Cues and to the Words. Resets the Cycle counter to 0 and increments the Trial counter by 1.

note: The activation of the model's units is reset to the resting level when the unit was created (via during model initialization or a subsequent auto-load), not to the global parameter, *rest*.

4.16 Script processor

command: S

purpose: Allows the user to enter the name of the script file used for automated execution of the model. See section 5 for details on the available script commands.

note: The file must have an *.iac* extension or else an error message will be generated.

4.17 Exit

command: X

purpose: Exits the IA Model program.

note: Any Auto-loaded words are lost and must be Auto-loaded again. The model does not save any modified parameters when it is exiting.

5 Script processing

The IA program includes a script processor which allows the user to automate an experiment and write the results to a *.csv* file for subsequent analysis by a program such as Microsoft Excel. Each command is given on a separate line and you can use your favorite editor to enter the script.

The available script commands are described in the following sections and are as follows:

```
b:    Run Blank Cycle
c1:   Set Cue1
c2:   Set Cue2
d:    Display Items
n:    New Word
pt:   Print Trace
r:    Reset
rc:   Run Cycle
rs:   Run Settle
t:    Run Trace
wt:   Write Trace
```

5.1 Run Blank Cycle

script format: ['b',*ncycles*]

purpose: Clear the external input to the letters and the cues and cycles the model for *ncycles* (an integer value).

note: This can be used to model the time interval between the presentation of stimuli.

5.2 Set Cue 1 (English) or Cue 2 (Spanish)

script format: [c1 (or c2)]

purpose: Set the cue's external input to 1 in order to activate the language.

note: c1 simulates setting the experiment participant's focus to their L1 (English) and c2 to their L2 (Spanish)

5.3 Display Items

script format: [d, [word | language | cue][,word | language | cue]]

purpose: Display activation of one or more words or languages entered at input prompt. Typically used at the end of the script to visual present results.

note: Must have run at least 1 cycle. Similar to Display 1 or More command (see Section 4.8).

5.4 New Word

script format: [n,*word*]

purpose: The given word is presented to the model's input.

note: Where *word* is a 3 to 5 letters and already in the model's lexicon.

5.5 Run Trace

script format: [t, type]

purpose: Write a trace record to the trace log file and append the *type*. A trace record contains: *[cycleno, input word, input word act, l1 act, l2 act, l1schema act, l2schema act, type]*. The *type* is a text comment you add which typically describes the type of event that is being logged.

note: The trace log is maintained internally and can be viewed by the Print Trace command, Section 5.6. It is written to a *CSV* file using the Write Trace script command.

5.6 Print Trace

script format: [pt]

purpose: Write all trace records created thus far via the *t* script command to standard output (i.e., the console).

note: Used to quickly view the trace records created without having to first write them to a *CSV* file and then running an external viewer.

5.7 Run Settle

script format: [rs, pool]

purpose: Runs the model until the pool settles (i.e., activation converges to value, as measured by a change below a small threshold).

note: Typically used to put the model's pools into a stable activation state before writing a trace record.

5.8 Write Trace

script format: [wt]

purpose: Write all trace records created thus far via the *t* script command to a *log.CSV* file for subsequent analysis by a program such as Microsoft Excel.

note: Adds the cycle number and the duration of the cycle to each trace record written as an analysis aid. See Section 5.10 which show the *log.CSV* file for the script example.

format: The first log record written is a *header* record, with text describing the log entries. Each subsequent log record contains the values described in the header record, in the same order and separated by commas. Each record ends with a newline character.

header = ['Type','Input','Word','L1','L2','L1 LDT','L2 LDT','No. cycles','Event duration']

5.9 Script Example

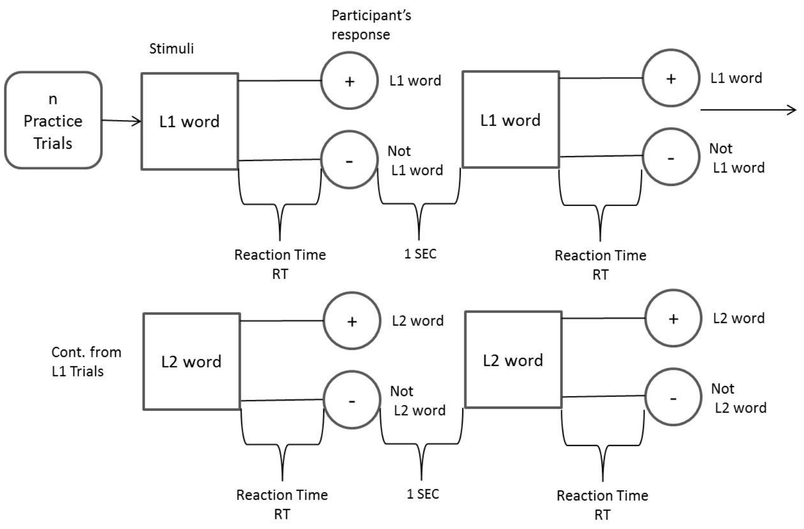


Figure 8: Experiment 1 procedure. After n practice trials, participants are presented with a letter string and asked to decide whether it is a string in either L1 or L2. Language switches occurred on alternating trials indicated by a change in color background.

The script example shown on the following page is based on Experiment 1 conducted by von Studnitz and Green (1997) as implemented in the computational model of Valenti and Scheutz (2013). In this study, bilinguals are asked to decide whether or not a presented letter string (may be a word or non-word) was a word in L1 or in L2 using an alternating runs paradigm (i.e., there is predictable switching between languages) and the color of the background on which the word was presented served as an external cue informing participants of the required language for decision.

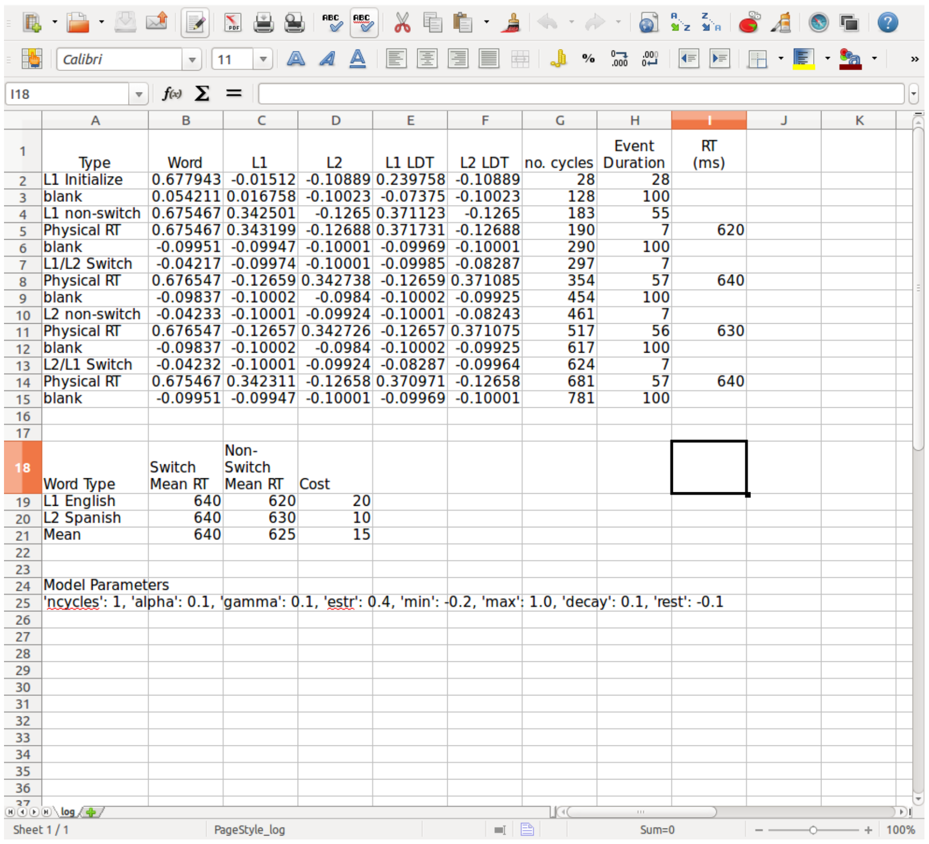
The experiment (illustrated in Figure 8) measured a participant's reaction time, i.e., from when a word was presented on the computer screen to when the participants press the "+" or "-" key. It is apparent that the reaction time (RT) consists of two components: the time it takes to activate the schema plus the time it takes for the participant to move his or her arm and press the key. For the purpose of the output data mapping, only the schema activation time is of interest and the remainder of the reaction time is treated as a constant. Thus in the model, only the number of cycles from the resting level of the schema until it settles at its activation level is measured.

On the following page is the script which runs a simulation of Experiment 1. The comments are not part of the actual script and are added to the example for documentation purposes only; they should not be included in any scripts. The original experiment conducted by von Studnitz and Green used German-English bilinguals; however the researchers do not claim their results to be language-specific. In our simulation we use English and Spanish words, achieving similar language switching costs.

Script for language switching experiment:

```
n,cat                      # present the word cat to the model
c1                         # set the cue to English
rs,cat                     # run the network until cat settles to an activation value
t,L1 Initialize            # log a trace record with the comment Initialize
b,100                      # run for 100 cycles with no input, i.e. a 'filler' trial
t,blank                    # log a trace record for filler trial
c1                         # set cue to English
n,table                    # present the English word table
rs,english                 # run until English settles on an activation value
t,L1 non—switch            # log a trace record with given label
rs,l1                      # run network until l1 schema settles
t,Physical RT              # log a trace record indicating a Reaction Time
b,100                      # run a filler trial
t,blank                    # log a trace record for filler trial
c2                         # set cue to Spanish
n,azul                     # present the Spanish word azul
rs,spanish                 # run until Spanish settles on an activation value
t,L1/L2 Switch             # log a trace record and label it a L1 to  L2 language switch
rs,l2                      # run network until l2 schema settles
t,Physical RT              # log a trace record indicating a Reaction Time
b,100                      # run a filler trial
t,blank                    # log a trace record for filler trial
c2                         # set cue to Spanish
n,vino                     # present the Spanish word vino
rs,spanish                 # run until Spanish settles on an activation value
t,L2 non—switch            # log a trace and indicate it is a non language switch
rs,l2                      # run network until l2 schema settles
t,Physical RT              # log a trace record indicating a Reaction Time
b,100                      # run a filler trial
t,blank                    # log a trace record for filler trial
c1                         # set cue to English
n,chair                    # present the English word chair
rs,english                 # run until English settles on an activation value
t,L2/L1 Switch             # log a trace record and label it a L2 to  L1 language switch
rs,l1                      # run network until l1 schema settles
t,Physical RT              # log a trace record indicating a Reaction Time
b,100                      # run a filler trial
t,blank                    # log a trace record for filler trial
wt                         # write all log file in CSV format
d,english,l1,spanish,l2    # display listed nodes in pyplot chart
```

5.10 Language Switching Experiment Trace File



Type	Word	L1	L2	L1 LDT	L2 LDT	no. cycles	Event Duration	RT (ms)
L1 Initialize	0.677943	-0.01512	-0.10889	0.239758	-0.10889	28	28	
blank	0.054211	0.016758	-0.10023	-0.07375	-0.10023	128	100	
L1 non-switch	0.675467	0.342501	-0.1265	0.371123	-0.1265	183	55	
Physical RT	0.675467	0.343199	-0.12688	0.371731	-0.12688	190	7	620
blank	-0.09951	-0.09947	-0.10001	-0.09969	-0.10001	290	100	
L1/L2 Switch	-0.04217	-0.09974	-0.10001	-0.09985	-0.08287	297	7	
Physical RT	0.676547	-0.12659	0.342738	-0.12659	0.371085	354	57	640
blank	-0.09837	-0.10002	-0.0984	-0.10002	-0.09925	454	100	
L2 non-switch	-0.04233	-0.10001	-0.09924	-0.10001	-0.08243	461	7	
Physical RT	0.676547	-0.12657	0.342726	-0.12657	0.371075	517	56	630
blank	-0.09837	-0.10002	-0.0984	-0.10002	-0.09925	617	100	
L2/L1 Switch	-0.04232	-0.10001	-0.09924	-0.08287	-0.09964	624	7	
Physical RT	0.675467	0.342311	-0.12658	0.370971	-0.12658	681	57	640
blank	-0.09951	-0.09947	-0.10001	-0.09969	-0.10001	781	100	

Word Type	Switch Mean RT	Non-Switch Mean RT	Cost
L1 English	640	620	20
L2 Spanish	640	630	10
Mean	640	625	15

Model Parameters
'ncycles': 1, 'alpha': 0.1, 'gamma': 0.1, 'estr': 0.4, 'min': -0.2, 'max': 1.0, 'decay': 0.1, 'rest': -0.1

Figure 9: Trace CSV file read into a spreadsheet, which calculates reaction time (RT)

The spreadsheet shown in Figure 9 was created to contain the trace records written by the script in Section 5.9. Notice that row 1, cols. A through H contain the header record automatically written by the *wt* command and rows 2 through 15 are the 14 trace records written to the log by the *t* commands in the script. Column I (RT) is computed by a formula in the spreadsheet; this is used to compute the statistics in rows 19 to 21. Thus it is easy to see how a spreadsheet can be set up in advance to analyze the data written to a *.CSV* file by the model's script.

6 Core Routines

In this section, the basic structure of the core routines used in the IA program model is explained.

6.1 Cycle (Model Control Loop)

```
#  Cycle(act_dataset) cycles through the pools, collecting the net input of each unit
#  and then updating the unit activation
#
#  Input: act_dataset is a list which contains a record of a pool's activation for
#         each update across cycles. It is hard-coded to the words pool.
#  Control variables: ncycles controls the number of iterations of the
#                     net input & update cycle
#                     verbose controls printing of each update cycle to the standard
#                     output (console)
#  Returns: act_dataset, act_langset appended with the last ncycles activation records
#           for each unit in the pool
def cycle_pool():
    global verbose, cycleno, act_dataset, act_langset, act_schemaset
    act_trial = []
    act_trial_lang = []
    # ensure ncycles is type int bc doSetParams converts it to float
    for reps in range(int(params['ncycles'])):
        cycleno += 1
        # gather netInput from pools
        netInput(cues)
        netInput(lets)
        netInput(words)
        netInput(lang)
        netInput(schemas)
        # update the pools
        update(cues)
        update(lets)
        update(words)
        update(lang)
        update(schemas)
        act_dataset.append(readActivations(words))
        act_langset.append(readActivations(lang))
        act_schemaset.append(readActivations(schemas))
        act_trial.append(readActivations(words))
        act_trial_lang.append(readActivations(lang))
        if verbose is True:
            print 'Word Activations:'
            print('Cycleno: ' + repr(reps + 1) + ' ' + repr(act_trial[reps]))
            print 'Language Node Activations:'
            print('Cycleno: ' + repr(reps + 1) + ' ' + repr(act_trial_lang[reps]))
    return act_dataset, act_langset
```

6.2 Gather Network Input

```
#   function netInput(rcvr_pool) parses the projections in rcvr_pool and looks up
#   activation of each sending unit
#
#   The standard netInput routine computes the net input for
#   each pool. The net input consists of three things: the external input, scaled by
#   estr; the excitatory input from other units, scaled by alpha; and the inhibitory
#   input from other units, scaled by gamma. For each pool, the netInput routine first
#   accumulates the excitatory and inhibitory inputs from other units, then scales
#   the inputs and adds them to the scaled external input to obtain the net input.
def netInput(rcvr_pool):
    global pool_list, params
    # generic pool function
    for key, unit_list in rcvr_pool.iteritems():
        for unit in unit_list:
            in_pool = False
            excitation = 0
            inhibition = 0
            if not unit.isProjNone():
                for sender in unit.getProjList():
                    #print repr(unit.getProjList())
                    from_keypos = sender[0]
                    from_key = from_keypos[0]
                    from_pos = from_keypos[1]
                    weight = sender[1]

                # check to see if key is in any sending pool, i.e. lets, words, lang
                    for pool in pool_list:
                        if from_key in pool:
                            activation = pool[from_key][from_pos].getActivation()
                            in_pool = True
                            break
                        else:
                            in_pool = False
                    if in_pool is False:
                        print("NetInput: Unrecoverable Error. No pool found.")
                        sys.exit(1)
                    if activation > 0:     # process only positive activations
                        if weight > 0:
                            excitation += weight * activation
                        elif weight < 0:
                            inhibition += weight * activation
            excitation *= params['alpha']
            inhibition *= params['gamma']
            unit.setNetInput(excitation + inhibition + unit.getExtInput()
                                            *params['estr'])
    return
```

6.3 Standard Update

```
# Standard update. The update routine increments the activation of each unit,
# based on the net input and the existing activation value.
#
def update(pool):
    global params
    # generic pool update
    for key, unit_list in pool.iteritems():
        for unit in unit_list:
            activation = unit.getActivation()
            if activation > 0:
                unit.setActivation(activation + (params['max'] − activation)
                * unit.getNetInput()
                − params['decay'] * (activation − params['rest']))
            else:
                unit.setActivation(activation + (activation − params['min'])
                * unit.getNetInput() − params['decay']
                * (activation − params['rest']))
    return
```

7 System Error Messages

The following errors are indicative of severe problems in the model engineering. As such they should be reported to the author, providing as much detail as possible.

NetInput: Unrecoverable Error. No pool found: There is a projection to a pool which does not exist.

8 Key Concepts in Lexical Decision Tasks

The lexical decision task (LD) is a procedure commonly used in psycholinguistics experiments, measuring how quickly participants classify stimuli as either words or non-words.

8.1 Lexical-semantic Processing Effects

This section reviews the most common effects reported in the literature associated with lexical decision tasks.

Word Superiority: When forced to choose which of two letters appeared in a given position in a stimulus item, subjects were more accurate when the letters occurred in words as opposed to non-words or presented in isolation. The latter result is notable since a single letter places a very low load on memory; it is difficult view the lower single letter accuracy as a result of forgetting. Thus, it appears that context plays a role in perception (J. L. McClelland & Rumelhart, 1989).

Stimulus quality: when a word stimulus is visually degraded, it increases the duration of most responses (White & Staub, 2011)

Semantic priming: is when a target word (e.g., SPOON) is identified more quickly when preceded by a related prime (e.g., FORK) compared with an unrelated word (e.g., CAT). A reduction in stimulus quality slows the processing of unrelated targets more than that of related targets (Borowsky & Besner, 2006).

Word frequency: in which high-frequency words are identified more quickly than low-frequency words. Stimulus quality does not differentially affect the processing of high and low-frequency words; just on the main RT (Borowsky & Besner, 2006).

8.2 Common Problems in LD Experiments

Discrimination between words and pronounceable (phonotactic) non-words: Is 'Quome' a legal word? LD tasks usually forces participants to interrogate their lexicon. If the stimulus pseudoword differs too much orthographically from the words' structure, participants may use strategies counter to the intent to probe lexical/semantic organization and processing dynamics. Thus the stimulus characteristics for words and pseudowords in an LD task should have the the same orthographic structure (Borowsky & Besner, 2006).

No consistent advantage between frequent letter clusters in pronounceable words: No advantage between *PEEP* or *TEEP* compared to much less frequent clusters such as *POET* or *HOET*.

No advantage for letters in a context that strongly constrains the identity of the letter For example, the *C* in *CLUE*: only three possible letters make a word in the context of *_LUE*. Alternatively, in contexts imposing much weaker constraints, the *C* in *CAKE*, there are 10 letters that make words in the context of *_AKE* (J. L. McClelland & Rumelhart, 1989).

References

Anaconda Python distribution. (2017). `https://www.continuum.io/downloads`. (Accessed: 2017-07-28)

Borowsky, R., & Besner, D. (2006). Parallel distributed processing and lexical–semantic effects in visual word recognition: Are a few stages necessary? *Psychological Review*, *113*(1), 181–195.

Dijkstra, T., & Heuven, W. V. (2002). The architecture of the bilingual word recogniton system: From identification to decision. *Bilingualism: Language and Cognition*, *5*(3), 175–179.

Green, D. W. (1998). Mental control of the bilingual lexico-semantic system. *Bilingualism: Language and Cognition*, *1*, 67–81.

McClelland, J., & Rumelhart, D. (1981). An Interactive Activation Model of Contextual Effects in letter perception: Part 1. *Psychology Review*, *88*(5), 375–407.

McClelland, J. L. (2015). *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises.* Stanford, CA: Stanford University. Retrieved from `https://web.stanford.edu/group/pdplab/pdphandbook/`

McClelland, J. L., & Rumelhart, D. E. (1989). *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises.* Cambridge, MA: The MIT Press.

Valenti, A. P., & Scheutz, M. J. (2013). A computational model of bilingual inhibitory control in a lexical decision task. In *The 12th international conference on cognitive modeling.* Retrieved from `http://iccm-conference.org/2013-proceedings/papers/0042/index.html`

von Studnitz, R. E., & Green, D. W. (1997). Lexical decision and language switching. *International Journal of Bilingualism: Cross-Linguistic Studies of Language Behaviour*, *1*, 3–24.

White, S. J., & Staub, A. (2011). The distribution of fixation durations during reading: Effects of stimulus quality. *Journal of Experimental Psychology: Human Perception and Performance*. doi: 10.1037/a0025338