Questions OB1

Approach:

DESIGN questions:

You want as output all the words that are activated. So when you read a word, you want to activate that word through bigrams, and all the words in it.

But, only 1 word above the threshold and is recognized. And other words are inhibited.

Need to turn off inhibition?

You are looking for a way to print all sub-threshold words as well.

Choose 3 most activated words at the time of recognition?

Or all words above half the threshold?

Or all the words? But then you really have a lot of them.. Because even words that are not in the target will be activated: bigram "la" will activate a lot of words.

A word is formally recognized when it reaches a threshold. BUT this threshold depends on length, frequency, etc.

Structure:

The new task must Simulate\_experiment?

Simulate\_experiments: Structure? In the beginning difference between Sentence or Flanker, but then no more?

Script seems to automatically do a word recognition task. Is it useful to write our new task in this script? Sentence and flanks are similar, but the new task is very different.

* (Why not bigrams possible of different words?)
* We get the words of Beyersmann, or not: But all that is needed is just a list of words.

First make your own list

Ultimate goal: Precisely replicate Grainger's experiment

In main.py: freq and pred are calculated based on only the words in the task. ? "PSC"?

In main\_exp.py is freq calculated based on SUBTLEX?

Fix: !!! parameters.py is used instead of paramters\_exp.py: in read\_saccade\_data, reading\_commmon, reading\_functions

Question: read\_saccade\_data line 249: where do these traffic jams come from?are they made by create\_freq\_pred\_files\_fr? Output not the same as already exists

Has errors:

-line 28: columns do not match the traffic jam!

-line 48: get words extracts words from the German PSC.. so 0 overlap.

Fixed

﻿Create\_freq\_pred, line 19: why do we use a threshold here?

For each language and task:

1.Stimuli/EmbeddedWords\_stimuli\_all\_csv is Texts/EmbeddedWords\_freq\_pred.txt?

->

2. Texts/EmbeddedWords\_freq\_pred.txt + Texts/Frequency\_english.txt is

Data/EmbeddedWords\_frequency\_map.dat

So EmbeddedWords\_frequency\_map.dat has the frequency of the most famous words of the language, and all the words of the task stimulus.

! Word inhibition matrix is False everywhere. Dictionary too small, parameters too high or coding error>]?

﻿You only have inhibition if the words are the same length!! our case, it results in BANK and BANKER in not having any overlap, and thus not having any inhibition on each other. Is this the goal?

What format should stimulus csv have?

Priming task:

In other tasks, eye position is always in the center of word.

How should it be modeled for priming?

Nadat de prime weg is, saccade blindness? Delay in eye fixation shift and attentional shift?

Target stimulus center in the same place as prime center?

I don't think there's a transfer between UPPERCASE and lower.

So you get uppercase bigrams, which don't match with lowercase words. This has now been fixed by making stimulus lowercase, but should this be built in?

Interesting:

OWNER activates ONE the most, because this word has a lower threshold

Time prime, time target etc. take over from Beyersmann

Beyersman 2020:

Also 88 non-words in stimuli.

Design:

Beyersman says: stimulus 3 seconds or until you give an answer.

Answering in our case: as soon as a word comes above threshold? Or this + reactive time?

Script doesn't necessarily have to break when answered: you can save RT and still run the cycles. You can also go to the next trial. Doesn't really matter

Next up:

Collect and plot all the data?

Make a separate script, ask Noor for her Jupyter notebook?

Design question for merging: Do you want to run multiple tasks in a row? Because then you have to import parameters several times.

Next task:

Design:

* Parameters:
  + 2 options:
  + Or you run 1 experiment at a time, then you only have parameters\_exp call 1 time.
  + Subsequently:
    - Or, in each sub script you call the right parameters based on a pickle? Not efficient
    - Or, you make parameters task-dependent:
      * Get\_params(task): if experiment, get the same
    - imports all parameters matrix into main, and passes all relevant parameters to the relevant functions.
    - Or you turn the parameters into a large data frame, pass it on to all subscripts.
* Main and main\_exp: Done
* Task attributes:
  + separate task attributes file,
  + Declare all task attributes in main?
  + Separate class in script? Like task. ncycles

Current structure:

Each task has its own attributes, and asks for its own parameters.

So:

At the beginning of main exp, global\_params is imported. It states in which task there should be run.

Then task\_attributes is called on the basis of which task it is: that is an object with all the attributes of the task to be ran.

Bvb: task\_attributes.language =’french’ if the task is Flanker

After that, task\_params is called, based on the attributes. This is different depending on whether the task is an experiment or not.

E.g. Task\_params.epsilon= 0.035

Then, instead of importing parameters\_exp.py or parameters.py again in each sub-module, which causes conflicts, parameters and attributes are arguments of the function that calls the submodules.

For example: instead of

Main.py:

Import parameters\_exp

Simulate\_experiments()

Simulate\_experiments.py:

Import parameters\_exp

->

Main.py:

Import parameters\_exp

Simulate\_experiments(parameters)

* Affixes list with frequency included,
* Activation via bigrams like everyone else

Simulate\_experiments:

Concatenation of individual\_words (stimulus of task) and freq\_pred\_dict to make lexicon.dat.

But freq\_pred\_dict should already contain all the stimuli.

Why this merge?

* Answer: freq\_pred\_dict are missing some words: words that were not in the SUBTLEX have been taken out. By individual\_words add it again ix the complete again.
* But the real question is: why is there a threshold in create\_freq\_pred?

Take into account eccentricity and edge weight? now: no

Bigrams replace position 1 and load? Interference with edge highlight. Or manually insert special bigrams with position None? Now: separate list special bigrams

Are the same affix extra inhibitions on each other? So also use the bigrams to make inhibition matrix?

-No problem

\_p and pe in pear have the same position and edge weight.

Pickle inhibition matrix

Word-freq-dict: append words not present with freq X

Do you make affixes a special class? I.e. no inhibition is calculated with regard to other words.

Recap: To implement recognition of affixes, the approach is to add affixes as normal words to the word lexicon. If an affix reaches recognition, the word is split into 2 parts: len of affix, word-(len of affix), and word recognition continues.

In order to accurately recognize affixes, y must be recognized only at the end of word:  
it therefore has a special marker, and is registered as y\_. Words also get \_.

Therefore, mummy would be \_mummy\_, and y\_ would match with the last bigram.

So, you must add \_first and last\_ to bigrams list, and calculate their relative weight.

You cant just add the \_ to the word, because \_second will also show up as bigram

So you must implant the two special bigrams in the bigram list, and manually get their weights and position via the same formula as the normal bigrams.

The best way to do this is to directly implement this in the function that calculates bigram positions and weights (special bigrams will therefore show up in inhibition matrix -> words with same affixes will have exaggerated inhibition).

Then, you have a list of bigrams which contains the special bigrams.

Then, when calculating bigram-to-word excitation, you must also add \_ to begin and end of stimulus.

And then everything should work as usual.

Question: In stringToBigramsAndLocations in Reading-common, why the complicated structure?

stringToBigramsAndLocations:

first letter will always be in 2, last in -3. Never before because there is “\_” and “ “

So, if first =1 and second=2, do the special start bigram

And if first =-3 and last =-2, do the special stop bigram

TODO: check inhibition matrix to see if all goes well with latest modifications

Find a way to get insight into the inhibition matrix

Week 1 2022

Question: if you simply put \_ in front and at the end of every word, including bigrams, would that work? For example, parity and affix y would be \_parity\_ and \_y\_.

* NO! the first \_ in \_y\_ would not match and y would not be detected. The affix is therefore a special bigram where it is represented with only one \_ , i.e. y\_.

Is it bad that there is a specific distinction between affixes and words? In terms of assumptions

Question: All words are used for building inhibition matrix, affix not included? So e.g. affix ment\_, should that be in competition with the word \_mint\_? Or is it a separate category without inhibition values? OK

Comment: when you make inhibition independent of word length, you don't actually recognize anything anymore, because everything exerts inhibition on each other. BVB 'we' inhibits 'weak', so that weak is not recognized if it is the target.

What to do, reduce length criterion, or omit but tweak inhibition values?

DONE: word-to-word inhibition shifted from -0.07 to -0.005 (was one of the previous values)

Comment: In separate cases, less is both an affix and a real word, with different freq values. So adding the affixes list to the lexicon might cause problems

* One option is not to add the \_ during splitting to bigrams, but before that. So the lexicon already contains words like \_hello\_ and ity\_. Then you have \_less\_ and less\_. So this implies that the lexicon that a person has distinguishes between those two versions of less.

Now: affixes are included in inhibition, inhibition again depends on length and \_less\_ and less\_ both exist in the lexicon.

If ok, then the mechanism of affixes and implementing underscores system is ready

Next step:

Build a mechanism that recognizes affixes and opens an extra slot for the stem.

Affixes change to suffixes. DONE: affixes = suffixes + prefixes, where prefixes is now empty

Logging module built

Idea for pickle inhibition matrix:

# OPTIMIZE: here, implement mechanism to check wether a matrix with specific parameters has already been done. If yes, import, if no, make it and save it.

# 1. get parameters.

# 2. open inhib\_matrix\_configurations to check whether the previous matrix corresponds to the parameters in this run

# 3. open the previous matrix if yes, run code below if no

# 4. save inhib\_matrix\_configurations with current params

* So you do not get a separate file for every matrix in the past, but only the most recent.

Test : testing with non-words + suffix: milken

Findings: depends very much on inhibition values. At high values, long words are never recognized. E.g. Importantness.

Absolutely is recognized, but not if the stimulus is absolutelyness

Both absolutely and ness as absolutelyness are recognized, but with very low inhibit values: 0.0005 (instead of 0.0165).

Y in absolutely is only recognized later.

Interestingly: interesting is only recognized when inhibition is really very low. Ly\_ is not recognized.

NOTE: if a word is not frequent in the 200 most, it gets a freq of 0. How does this affect recognition?

TODO: How to stabilize inhibition waardes? Via optimize functions of the code?

TODO: Beyersmann 1: do experiment with truly-suffixed, pseudo-suffixed and non-suffixed words. Plot results with Notebook

Beyersman 2: difference in affix recognition between German and French.

TODO: find affix frequency list, for French and German.

Check inhibition matrix: Everything ok

All negative values of inhibition matrix?

Theory: CORNER and HIGHER both prime their stems. But, CASHEW does not prime CASH! So although CASHEW activates the bigrams of CASH, the inter-word inhibition ensures that CASH is inhibited exactly as much.

But how do you explain the priming of CORNER and HIGHER? There is something of a higher order process that discriminates between real-suffix and non-suffix, that the stem primes. How can the system built into OB1 provide priming? Words haven't split during prime yet, so that extra module has no effect.

Since in the case of CORNER and HIGHER there is splitting of stem and suffix, search specifically for the word

literature shows that word recognition takes about 500ms. So not necessarily too slow, maybe too fast?

-max bigram difference adjust

max\_threshold can also go down

\_Inhibition results: most robust at medium activation and low inhibition. With more intense excitation, too many random words. At higher inhibition, no long words recognized. If you maintain the balance, you can increase both exciation and inhibition. The same words are then recognized, but faster. For fixed inhibition and excitation, bigram-to-word inhibition has the effect that fewer random words above the threshold.

* Stable results at excitation = 2. 8, word2word\_inhib= 0.0006, bigram2word\_inhib= -0.14

At excitation = 6, bigram2word\_inhib = -0.05 and word2word\_inhib = 0.008, affixes are recognized fairly consistently in the first 2 cycles. But, is quite unstable: absolutelyness and importantness are just recognized or not recognized (never, not even with 30 seconds of stimulus). , although the word has been added to the lexicon. In addition, many other words also become active.

(importantness and absolutelyness first had a freq of 0, now minimum of dict).

Notebook works again!

Comment: the prime is now immediately recognized, so reaction is recorded immediately and all trials are wrong. So you actually want a mechanism where first only the affix is recognized - > searches for shorter word, and then the word is recognized. So you have to find a way to make affixes much more recognizable than long words.

1. Affixes should not have a word-to-word inhibition. word-to-word at 0 for all words.

* Only 0 inhib on each other, or on everything, or only on words? So can \_less\_ inhibit less\_ and vice versa?
  + Now: affixes have no inhibition on words and vice versa. But affix on affix, and word on word
* Easiest way to do this is change overlap between affixes

1. Lower threshold for affixes: double frequency for affixes.

Comment: then a lot of affixes come up, even age\_ for tighten.

With low threshold + no inhibition: preference for short words.

IDEA: inhibition weighs heavier on long words, because you have more overlap. You could normalize this by dividing the total amount of bigrams+monograms.

* Check if this is not already happening.
* No, but excitation actually depends on length. So if inhibition depends on length, it should approximately balance itself. But you have little control over that. So you can normalize excitation, and inhibition nornmalizing: long words will no longer cause problems.
* Or weighing inhibition with activation of words?

Question: threshold function: see reading\_functions

Chart, histogram

Description automatically generated

Bigram-to-word inhibite: comment: the effect is linear (every word receives same inhibition), so the max word does not change. However, effects can occur on which word reaches threshold first. -> high value: words with low threshold and low activations get disadvantaged.

question: cumulative activation function?

Threshold may be changed: up to 0.6

Thresholds of affixes are particularly low

Bigram gap change: lower -> higher contrast

To change:

Totalcycles, primecycles, targetcycles (make dynamic)

? unitActivations (divided by x?)

pm.bigram\_to\_word\_inhibition

pm.bigram\_to\_word\_excitation

pm.word\_inhibition

pm.decay

CYCLE\_SIZE

Remove double frequency

We need to find a good way for CORNER to activate CORN.

1. When word in CORN + ER is split, CORN boosts in activation at word level?
   1. You then get the problem that WRIT+ER will activate the word WRIT, and that does not exist
2. On recognition of affix, lower threshold for lengths of stimulus-affix. For x cycles.
3. Alternative: focus attention more on the side opposite affix? Then you do not get the effect that all non-related 4 letter words also become active.
4. Alternative 2: Activate bigrams specifically from the stem. So reactivate all bigrams without the letters of the stem? (aka. When word in CORN + ER is split, stimulus of CORN and ER twice as salient? So, increase excitation specifically for those bigrams? -> CORN activation at bigram level)
5. When word in CORN + ER is split, no more inhibition of CORNER on CORN? Or, inhibiting the word CORNER on 0 for everything?
6. «**Morphological decomposition based on the analysis of orthography":** Each affix is accompanied by a group of stems that often occur with it. Recognizing an affix, primes all those words (perhaps overlap with active bigrams) > allows for additional activation of those words. An extra (large) assumption
   1. No this is already not working.
7. Recognizing affix makes the affix 'invisible' - > prime and target match 100%?
8. Reduce inhibition between CORNER and CORN (since the beginning). So, all words with affixes have no inhibition on their stem.
   1. But, what about affixed words that don't exactly have a stem (RELATE-RELATABLE): if you search for RELAT, you won't find anything.
   2. Words you don't know? MILKEN-MILK. Or even SARB-SARBY. Do you expect priming? Inhibition has been learned, but is that realistic with a word you don't recognize?

2 Theories: 1. There is a mechanism that recognizes automatic affix and parses -> in this case you also expect a difference in words that you do not know. MILKEN will prime more than MILKEW. 2. No, you learn manually between each known word and his stem, that they should not be in competition with each other. In that case, CORNER will prime, but MILKEN will not.

<https://doi.org/10.1177/1747021819896766>: no priming for affix vs non-affix nonwords.

<https://link.springer.com/article/10.3758/s13423-011-0120-y>: priming for affixed nonwords, but not non-affixed nonwords!! (40ms prime)

If you decide to make the inhibition matrix 0 for words like CORNER-CORN, the model will work, but how do we interpret this? You don't learn to inhibit the stem for each pair, even if the stem has nothing to do with it? Nothing happens "mechanically" anymore that takes care of the priming effects, but everything depends on the matrix, which we build by hand.

Non-primed and primed reaction times closer together: you want the priming of related prime to be canceled by word2word inhibition, and then saved by special affix system: see point 4 in red above.

It is still very useful if we make the time span shortr:

* In more than 2 cycles it is easier to model complex effects
* At the moment it is often cycle 2: recognize affix, cycle 3: recognize word. Little resolution in score differences.
  + Doesn't matter, because how we design the affix system now implies that affix only exists to be recognized, and then all words become lower threshold. So that effect will happen regardless of whether the prime is still in the picture or not.
* Note: the whole "search words with length-length(affix)" is no longer necessary. If recognizing the affix of lowering the threshold of the remaining words is enough.
* comment: non-suffixed + suffixed are now the same in terms of RT. But, suffix is often recognized instead of word, because low threshold. FOR EXAMPLE: 'tion' in 'edition', then stimulus is 'edit'. 'tion' lingers and is highly activated word. Or even, 'ant\_' instead of 'cat'.
  + If you make affixes threshold hoger, you have the problem that they are not recognized in the first rounds .
  + Simply, «if word not in affixes”? So affixes cant be recognized at all, because they are not words
* In the first instance: effect of lowering the threshold, ensures that a lot of words exceed the threshold. Unfortunately, then usually the good one does not become the most active word.
* TODO: plot, get idea of balance word2word inhib and bigram2word inhib.
* Oscillation in inhibition values? Yes, inhibite values depend on the activation of the round before: first high activ. -> high inhib -> low activ. -> low inhib etc.
* Current problem: short words have too low threshold and too high activation compared to the rest.

Comment: with our simulation you would consider that «grassy» , «flower» primes, because recognizing an affix primes all 5 letter words. Is that realistic?

Comment: inhibition matrix is defined in the future by the model of how children learn to read. So the current model is not yet based on a 100% immutable theory. So we can change this inhibition matrix without too many assumptions, as long as there is a good rationale behind it, which remains consistent with our idea of how this matrix arises.

Comment: our recognition of affixes is independent of his stem, but is that true? Would you expect segmentation on SMARTPHONE? does SMARTPHONE prime SMART or PHONE?

??? problem with stimuli??? Speaker is under non-suffixed..

Answer: yes, but under unrelated: so it doesn't matter which prime, there is no overlap anyway

!tricky: play is seen as pla + affix -y. Action is seen as ac + tion. Ac has an edit distance of 2 with lots of words.

* Lcs met distance 1, onder 4 letters -> distance 0

! until now, every word in matrix always had inhibition in itself: WEAK inhibits WEAK. Take it out?

Next time: looking at TODO's for improvements, analysis results affix\_system on/off

Hint: use wordfreq package? (<https://pypi.org/project/wordfreq/>).

+ spacy or nltk, able to split words by affix + stem. Use algorithm instead of LCS?

Frequency:

HAL is text corpus

MorpholexFR – Morpholex equivalent

Morpholex extracts frequency from English Lexicon project

!!! Zipf definition:

it equals log10(frequency per billion words). So affixes.py was not accurate.

SUBTLEX-UK is Zipf

English Lexicon project is based on HAL frequency. But, that is an absolute value, not relative to corpus size.

French lexicon project adopts values from Lexique383, which are in the frequency per million (or log10 of it).

SO: first, convert MorphoLex-EN from HAL to Zipf, compare with current affix values.

Then Morpholex-FR to Zipf.

Problem, no one knows exactly how big HAL is. About 130-400 million.

Problem: Morpholex-EN zipf converted does not match with current affix values: it is always over it. -> so HAL must be bigger than 130 million, closer to 400

For next time:

MorpholexFR and MorpholexEN are used to measure affix frequency. But! Scales are not the same. Morpholex EN is based on HAL (rough word number – convert to zipf), and FR I don't think! Figure out which scale, and then convert to Zipf. English word dict (based on SUBTLEX-UK) is also in zipf, check if SUBTLEX FR is also in zipf. (it isnt)

Current state:

The freq and pred files are now converted to Zipf for EmbeddedWords, Sentence and Flanker (so, English and French). The affix frequencies of English are also in zipf scale. However, the French affix frequencies are still unclear! Check emails and correct

Next: check results if everything is still stable after modifying frequency scale.

Idea: cross reference THE in other databases to measure HAL size

Ok! So: Morpholex FRassumes f req values of FLP, found on Lexique.org. FLP takes over these values from Lexique383. These are in frequency per million.

HAL values compared to large online word database (1 billion)

* HAL is estimated at 400 million

Tests: SUBTLEX UK and Morpholex values for "hi" are the same in Zipf (if we assume a HAL size of approx. 400 000 000): about 5.3. In the final freq\_pred files, these values are also the same. So the affix freq values in English are correct.

For freq values, it must be clear what exactly we want to use: frequency of the lemma, or of the specific word? At the moment it is the word itself for English, but SUBTLEX also contains values for lemma frequency.

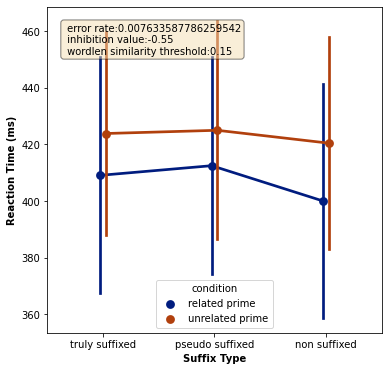
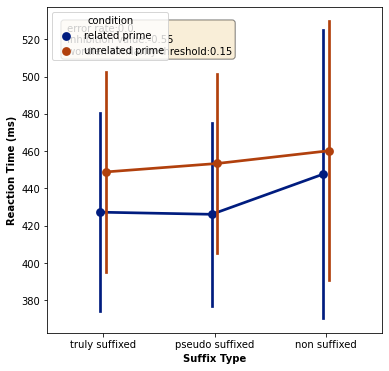
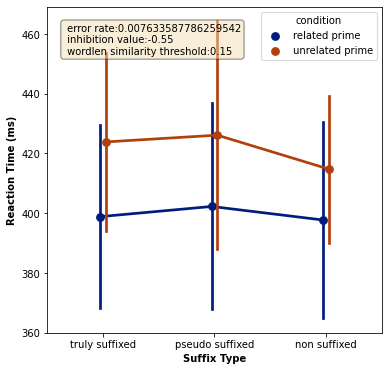
Lexique.org has frequency values in fpm, and the values correspond 1on1 to the MorphoLex values (and also to French Lexicon Project values). The current pred\_freq frequencies also correspond to Morpholex. Same as for English, the current word frequencies are of the word itself and not of the lemma.

TODO: add explanation in README. Cleanup files

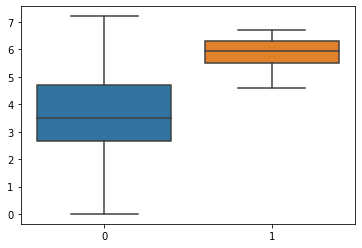
Update: You don't expect a priming of CASHEW on cash, but we had it. more Inhibition didn't work!! With more inhibition, no priming at all. Strange, because inhibition of CORNER on CORN is always at 0 (double check). How is that possible? So GIRLDE primes CORN as much as CORNER

Was due to affixes: their values are nu higher (from 6 to 7.21) (but, max is 7.67: \_the\_).

Current problem: with changing the affix values, the no-priming effect of non-suffixed disappears:



New affix values Old affix values No affix system, for reference



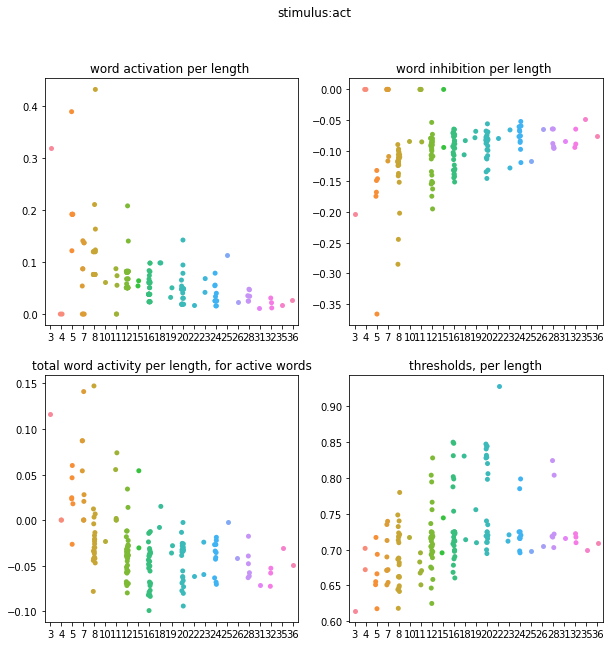
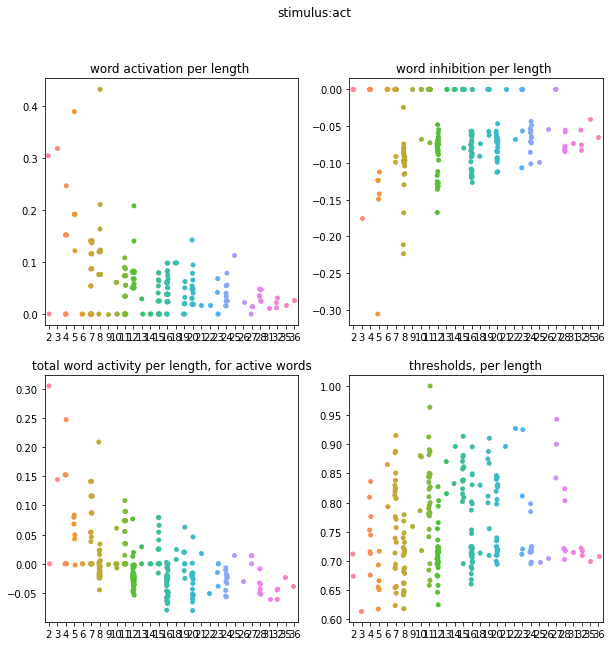
New affix values old affix values

Affix frequency distribution

Old values (orange) are on average higher, but maximum is lower.

When the new values are used, in 1 case fluffy is recognized too quickly while it is the prime.

New values ensure higher activation?



Comment: lexicon\_normalized\_word\_inhibition = (

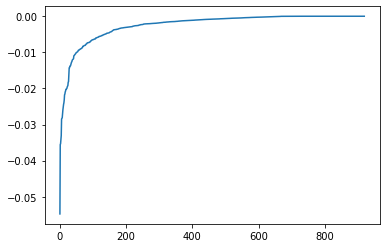
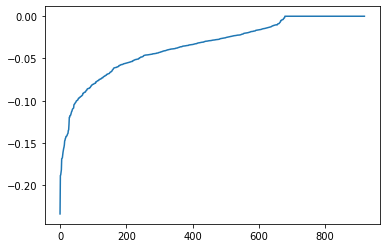
100.0/LEXICON\_SIZE) \* pm.word\_inhibition

Inhibition is divided by total lexicon. Lexicon is now much larger.

Next step: keep only suffixes that occur in Beyersmann.

Question: inhibition of all affixes is 0.0. is that realistic? No inhibition between affixes?

Idea: That weaken, weak does not inhibit, makes too small a difference. At the moment, if 100 words are active, almost all words in the lexicon are inhibited. (100 words have at least 1 letter overlap with everything). Make this inhibition more focused? Reduce the bigram gap? Or make the inhibition function an exponential? That only high overlap + activation values really inhibit?



Normal curve -(squared)

Doesn't work well: bias towards short words. (these have less overlap with others, so

Merge Beatriz:

Wordfreq from NL has been added. What is the source of that?

The lexicons used for the French and for the Dutch reading simulations were collected from a freely-available corpus of words in either language. We gathered the 2000 most frequent words from the French Lexicon Project (FLP) (Ferrand et al., 2010) to compose the French lexicon. Using the tool TreeTagger (Schmidt, 1994), we annotated the TCOF corpus (Benzitoun et al., 2012) with POS information, thus getting us the grammatical category of each French word comprising the corpus. Later, we used the SUBTLEX-NL (Keuleers et al., 2010) corpus to gather the 2000 most frequent Dutch words and create the Dutch lexicon used for the simulation. Finally, we used GECO (Cop et al., 2017) to gather the POS information on each Dutch word which composes the sentences in this corpus.

And how was the grammar predictability of EN and FR added? Was POS in FLP and from that you can deduce predictability?

Now you have French\_Lexicon\_Project.txt (old) and french\_lexicon\_tagged.csv (new) and french\_lexicon\_tagged2000.csv (new). Freq is averaged from books and movies? Same values, so txt replaced with new CSVs? And customize scripts.

Stimuli with POS: are they stimuli or texts with information about stimuli?

ToDo: look at difference of GetResults and GetResultsSimulation in analyze\_data\_pandas

For clarity:  
New tasks added: Classification (dutch) + Transposed (fr)

POS info on French and dutch, specifically for 2 tasks mentioned above + Sentence task

Still to do:

Create\_freq\_pred files: choose the right database and integrate POS.

Simulate\_experiment: use\_grammar\_prob

Analyse results: get\_results of get\_results\_simulation?

It should be:

Pred files are created per task in create\_freq\_pred files. Then, in simulate experiments, they are called with get\_pred\_files function from read\_saccade\_data, just like freq files

What about predictability of affixes? Avg of pred? 0?

from read\_saccade\_data, just like freq files

System for slots matching implemented? If yes, interaction with affix system (create new slot for recognized stem)

How are the POS bef, ProbBef etc. made? Did these files already exist through another experiment, or handwritten?

Subtlex EN: CD is number of documents the word appears in. Max 4, so not the same measure as for other languages!

But above all: POS prob matrix I understand (is generally defined per language), Stimuli with POS also (simply matching Subtlex words with their POS),

but how is the task POS file created? (both POS\_prob\_{task} in Texts and Sentence\_{task}\_POS in Stimuli)

Get\_pred\_freq can go to create\_...

Question: Subtlex NL already contains POS, why TreeTagger uses? And why is there dominant.POS and POS colums?

Normalize\_pred\_values in reading\_functions?

Next: replace lengths\_to\_be\_matched with code martijn

Consistency checks in beginning

Get\_threshold: Comment says "compute thresh based on freq and pred", but pred is not given as input?

Pred\_values is ambiguous: it is mentioned in the code for each task, and according to the code should be used to determine the threshold. Only, pred\_value is not used by the threshold function. The new tasks, which do use pred\_values, have a completely different structure, and do not use the threshold function at all. So what about those pred values? Remove from the code for non-POS tasks? i.e. not co-determining threshold? (it isnt now anyway).

Interaction slot-system and affix: what should happen if an affix is recognized? Do all slots also have to accept a word of (word – affix) (slots 4,5,6 also accept 2,3,4)?

So: affix recognized. You first test slot 0. There you look for a word of the length of the stimulus. But, that same slot can also match a word of stem length.

Or maybe no interaction at all? At the moment we only want weaken to prime, but that has been done through the inhibition system. Priming therefore takes place independently of recognition. So is it necessary to be able to match the stem on a slot?

In any case, you do not want this mechanism to recognize WEAK as if it were fitting length with WEAKEN. There is priming, but no substitution.

At the moment, the code sets the activation of a word to 0 if it is recognized and matched. This ensures that a mismatched word gets out of the way and the good word gets a chance. But you can never really do it wrong, if it is not the right word you continue until the good one is matched. Is that correct?

It also ensures that the word disappears from activity immediately afterwards. With priming, that can be a problem, because if you set activity to 0, you will hinder priming.

And if you have the same thing 3 times? WEAK WEAK WEAK? Lock WEAK while keeping activation?

Maybe cooldown period of 2 cycles where he is active but can't be matched?

It is also the case that with the slot mechanism you make exponentially many mistakes the longer the sentence is. Very often words mismatched. Ie.” My name is quintijn “-> quit is matched on the slot name.

Lijn 866: also check if the target is in the right slot?

Lijn 646: # Now create list that will hold the recognized words

this list should reset with new stimulus?

Match\_slot\_check is now in the cycles, because it has to be recalculated when the stimulus changes.

In general: POS and pred\_values are intertwined: must be used more clearly in the code , but for that I first have to get an answer from Beatriz.

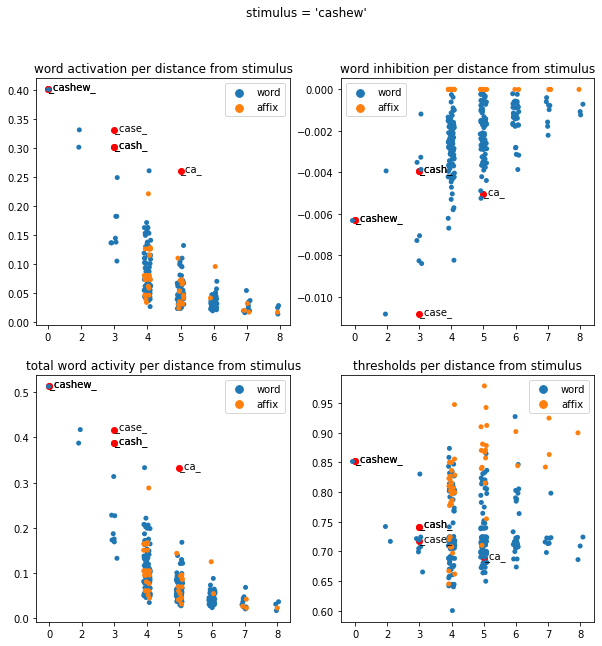
Inhibition values problem:

Recap:

You expect that with the affix system (affixed words and their stem have no inhibition on each other) that affixed words followed by their stem will be recognized earlier (priming effect) but not words that are not affixed.

However, this does not happen: there is hardly any priming.

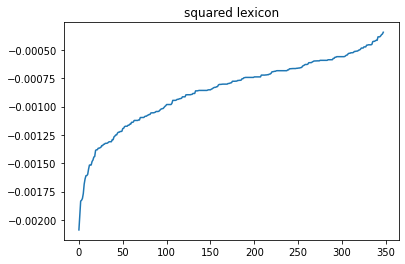
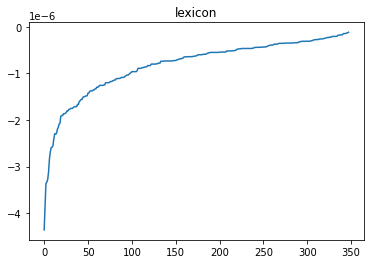
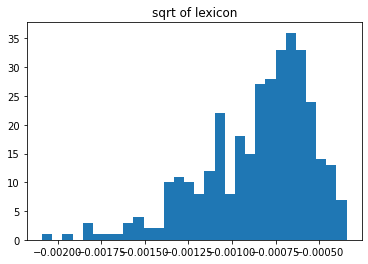
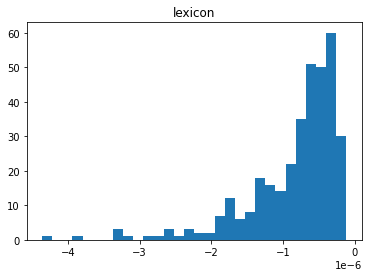
Test: activation higher with similarity bigrams

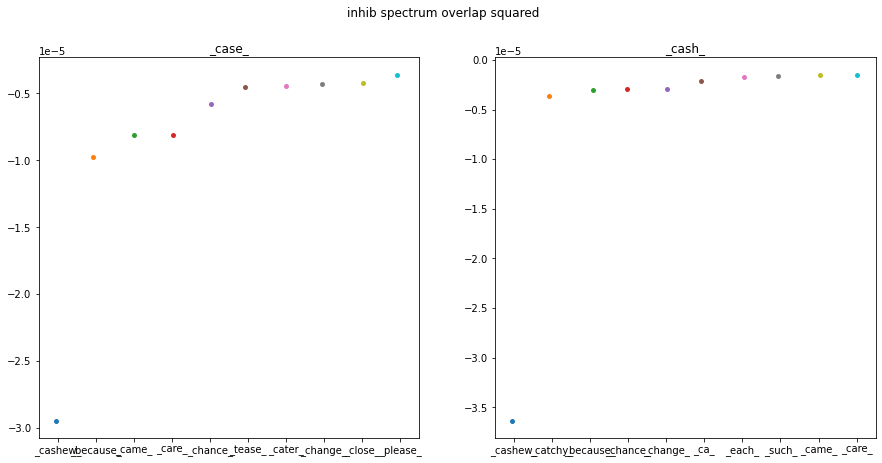
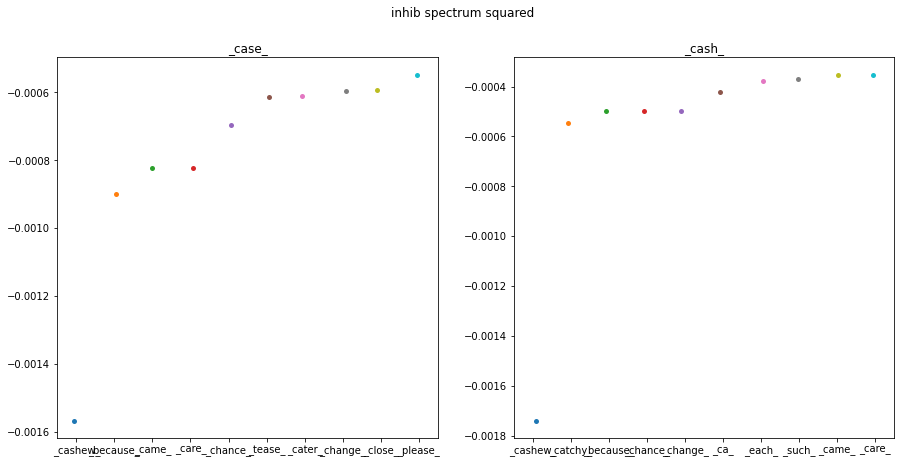
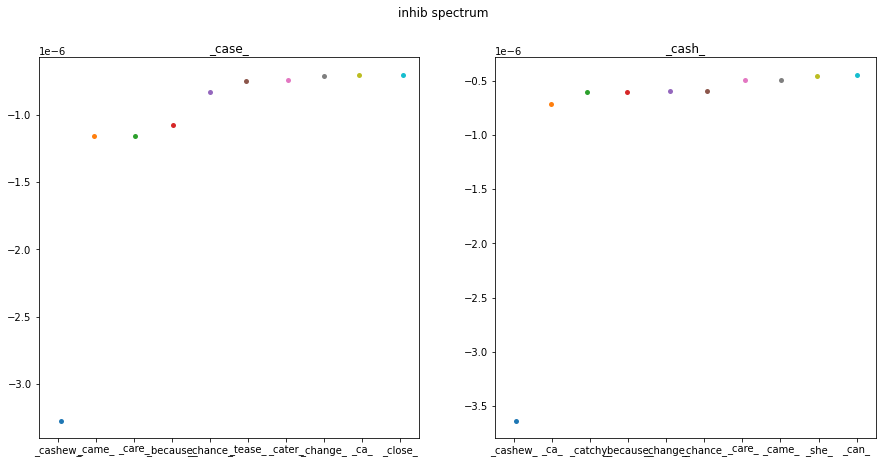


1. Activation / inhibition decreases with distance from stimulus
2. Cash still has a high activation, although we wanted cashew to inhibit Cash very much.
3. Why does Case have much more inhibition than Cash?

Todo: plot for cash and case which words are most responsible for inhibition

Lexicon squared:

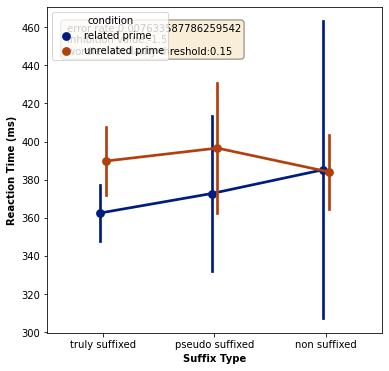
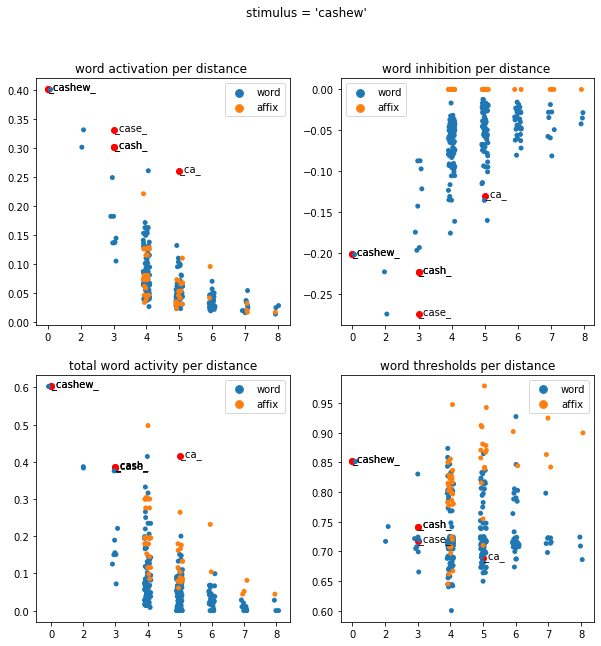


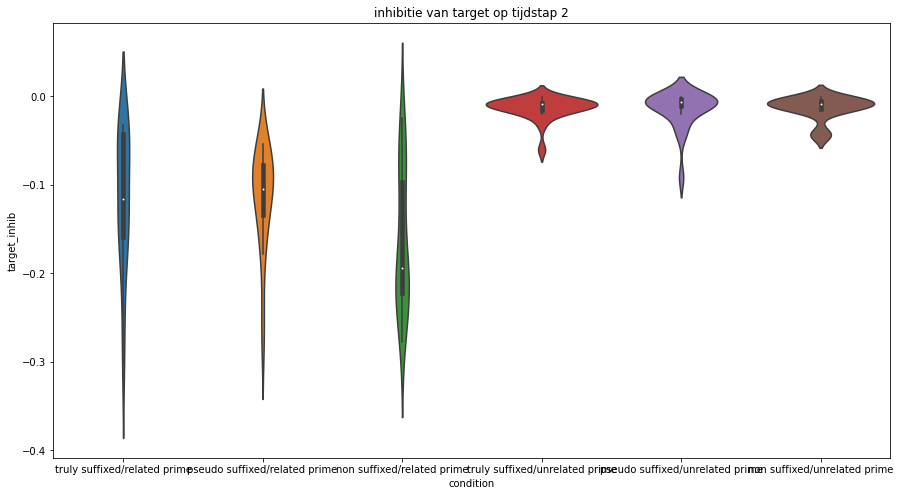


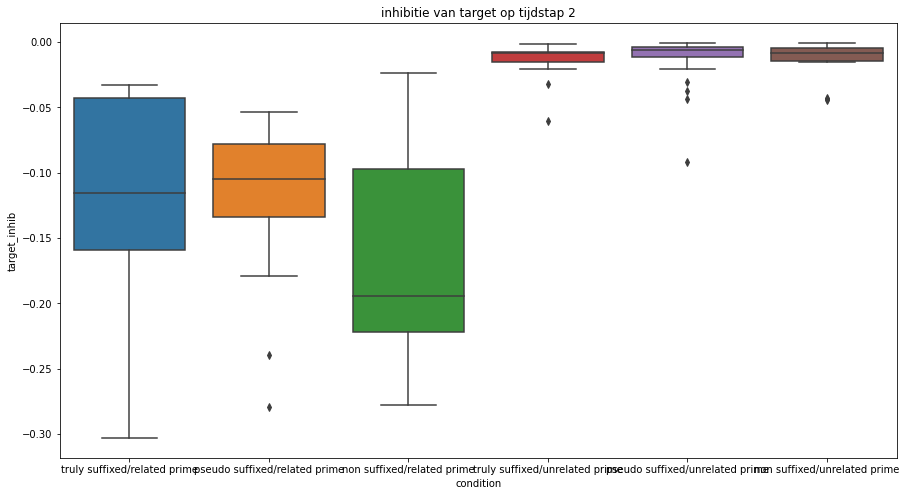
Next up: looking at squared + inhibition cutoff

Results:

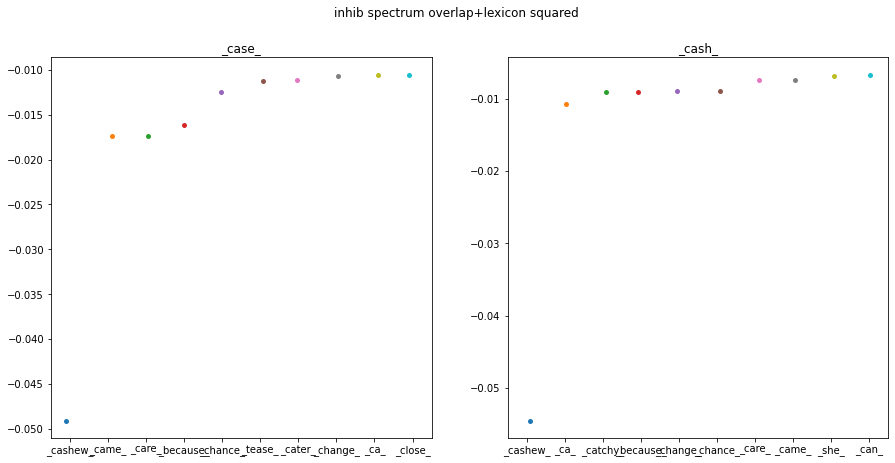
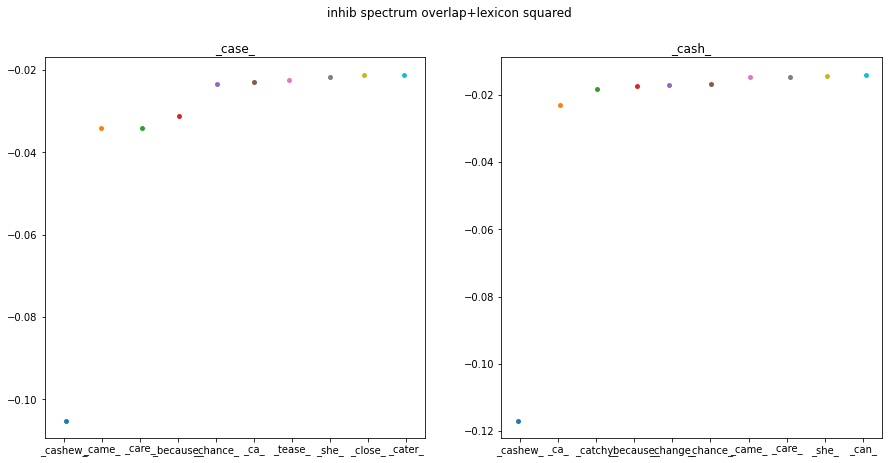
Inhibitie squared:





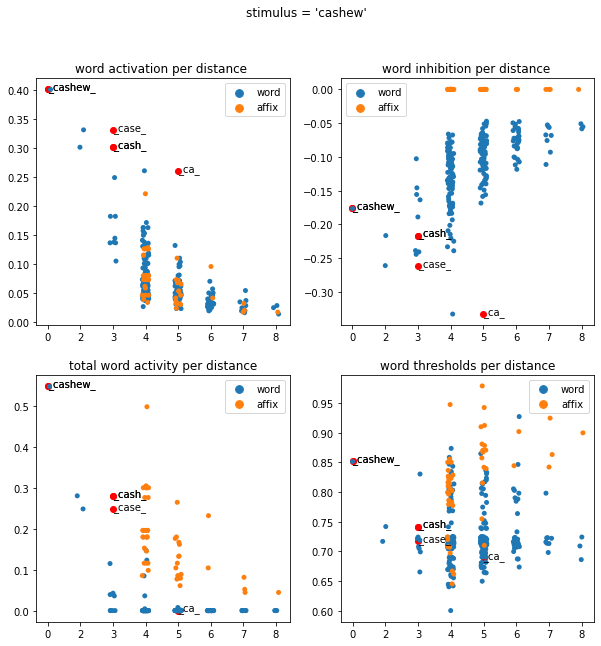
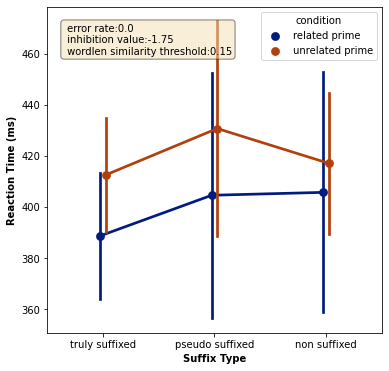
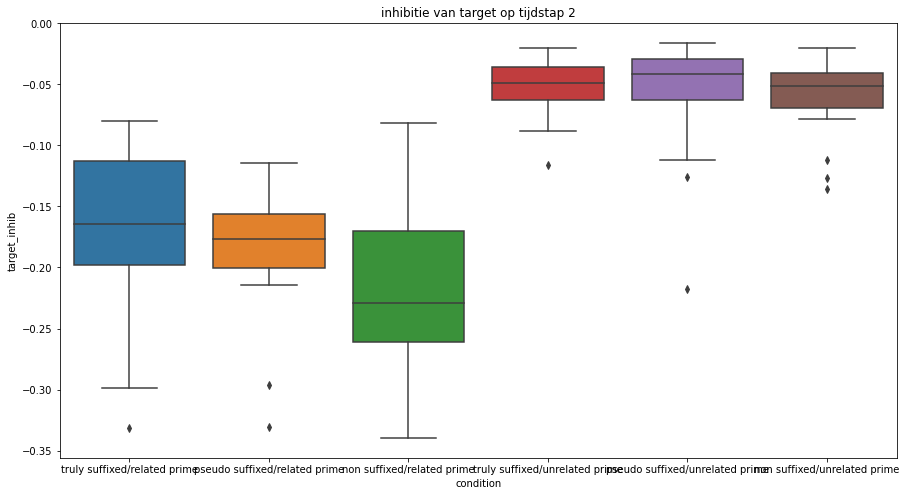
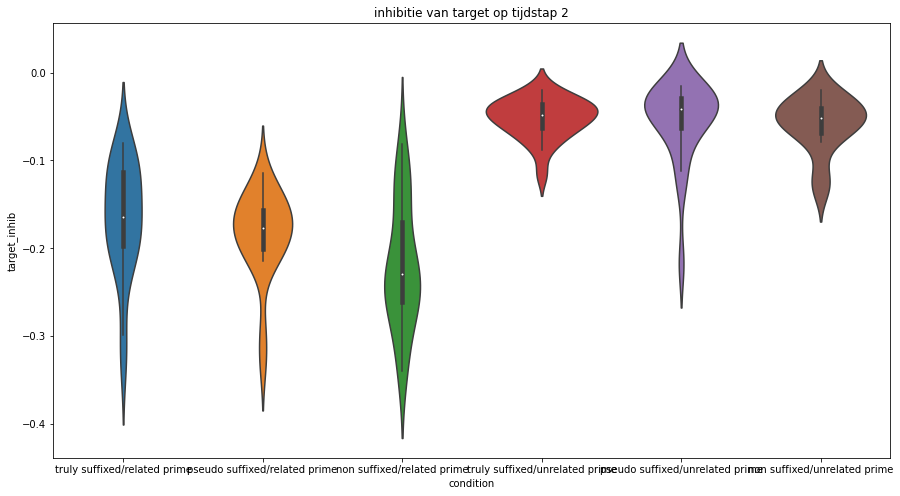


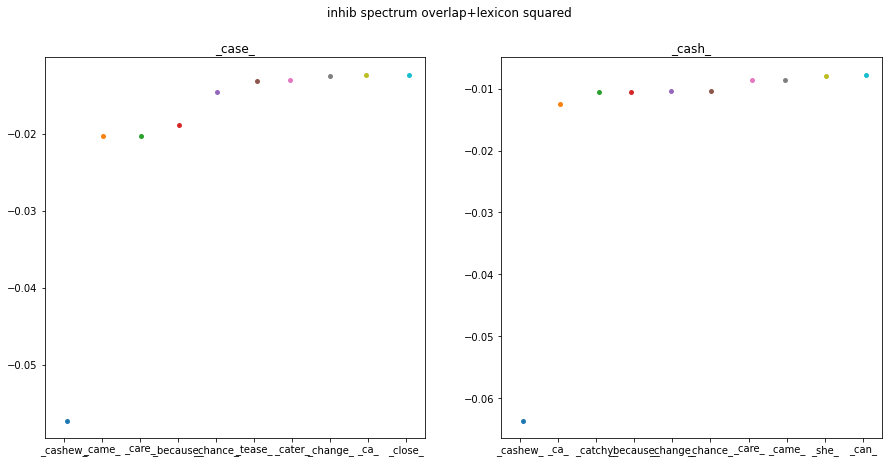
timestep 1

timestep 2

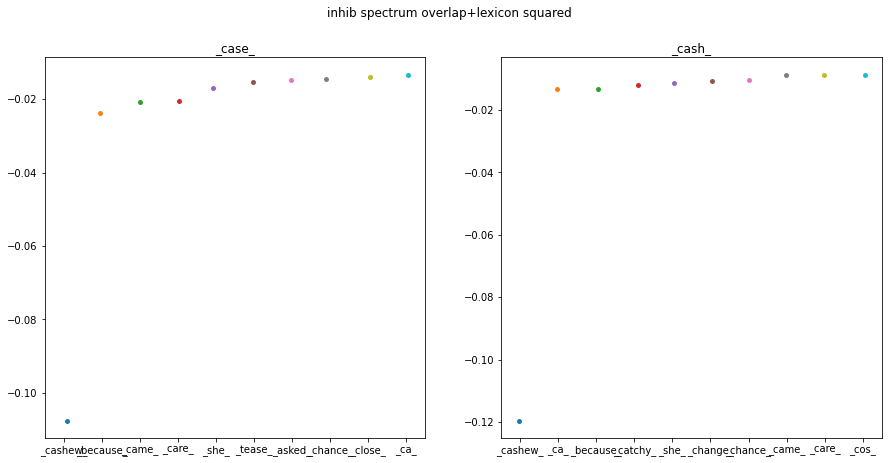
Note: at higher inhibition values, specifically for related prime / non suffixed, some words are never recognized.

Inhibition cutoff:



timestep 1

timestep 2



To consider: Dragon – drag is not exactly non -suffixed: -on is a valid suffix, and therefore OB1 calculates no inhib. Between dragon and drag

Cutoff: change to numerical cutoff instead off. 10 first words.

Konstantin:

I added the analysis files to the GitHub. The main notebook to analyze Ob1 results is OB1\_taskperformance.

Martijn mentioned that one of your first tasks will get to make normal text reading in German work again. Indeed, we have recently mostly focused on experiments, and changed many parameters, which probably has an impact on normal text reading.

At the moment, you will see in parameters.py, that experiments and PSC (text reading), use separate parameters. So this might actually help, because it mitigates the changes we have made to the code to make experiments work. Ideally, in the future, these parameters would be the same for text reading and experiments.

The main thing that worries me, is that reading and experiments used to be two different scripts. Martijn asked me to merge these, so now they both run in simulate\_experiments.py. But, it is therefore uncertain if PSC still works. Nobody has really tried it in years!

Perhaps, by diving deep into the GitHub previous commits ( maybe up until the point where I forked the code from Noor), you can see how the code used to look like. That might help.

I have also updated the Readme.md slightly, It gives a few hints on how the code works and how to install it properly.