Vragen OB1

Aanpak:

DESIGN vragen:

Je wilt als output alle woorden die geactiveerd zijn. Dus als je een woord leest wil je via bigrams dat woord activeren, en alle woorden daar in.

Maar, er komt maar 1 woord boven de threshold en wordt herkend. En andere woorden worden geinhibeerd.

Inhibitie uitzetten nodig?

Je zoekt een manier om alle sub-threshold woorden ook te printen.

3 meest geactiveerde woorden kiezen op het moment van recognition?

Of alle woorden boven de helft van de threshold?

Of alle woorden? Maar dan heb je er echt veel.. Want zelfs woorden die niet in de target zitten, zullen geactiveerd worden: bigram “la” zal heel veel woorden activeren.

Een woord is formeel herkend als die een threshold haalt. MAAR deze threshold is afhankelijk van length, frequency, etc.

Structuur:

De nieuwe task moet in Simulate\_experiment?

Simulate\_experiments: Structuur? In het begin verschil tussen Sentence of Flanker, maar daarna niet meer?

Script lijkt automatisch een word recognition task te doen. Is het wel handig om onze nieuwe task in dit script te schrijven? Sentence en flanks zijn namelijk vergelijkbaar, maar de nieuwe task is zeer anders.

* (Waarom geen bigrams mogelijk van verschillende woorden?)
* We krijgen de woorden van Beyersmann, of niet: Maar alles wat nodig is is alleen een lijst woorden.

Eerst eigen lijstje maken

Uiteindelijke doel: Experiment van Grainger precies repliceren

In main.py: freq and pred worden berekend op basis van alleen de woorden in de task. ? “PSC”?

In main\_exp.py wordt freq berekend op basis van SUBTLEX?

Gefixt: !!! parameters.py wordt gebruikt ipv paramters\_exp.py: in read\_saccade\_data, reading\_commmon, reading\_functions

Vraag: read\_saccade\_data lijn 249: waar komen deze files vandaan?worden deze door create\_freq\_pred\_files\_fr gemaakt? Output niet hetzelfde als er al bestaat

Heeft errors:

-lijn 28: columns matchen niet met de file!

-lijn 48: get words haalt woorden uit de duitse PSC.. dus 0 overlap.

Gefixt

﻿Create\_freq\_pred, lijn 19: waarom gebruiken we hier een threshold?

Voor elke taal en task:

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

->

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

Data/EmbeddedWords\_frequency\_map.dat

Dus EmbeddedWords\_frequency\_map.dat heeft de frequency van de meest bekende woorden van de taal, en alle woorden van de 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?

Welk format moet stimulus csv hebben?

Priming task:

In andere tasks is eye position altijd in center van woord.

Hoe moet het gemodeleerd worden voor priming?

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

Target stimulus centrum in dezelfde plek als prime centrum?

Volgens mij is er geen transfer tussen UPPERCASE en lower.

Dus je krijgt uppercase bigrams, die niet matchen met lowercase woorden. Dit is nu gefixt door stimulus lowercase te maken, maar moet dit ingebouwt worden?

Interessant:

OWNER activeert het meest ONE, omdat dit woord een lagere threshold heeft

Tijd prime, tijd target etc. overnemen van Beyersmann

Beyersman 2020:

Ook 88 non-woorden in stimuli.

Design:

Beyersman zegt: stimulus 3 seconden of tot je een atwoord geeft.

Antwoord geven in ons geval: zodra een woord boven threshold komt? Of dit + reactive tijd?

Script hoeft niet per se te breaken bij antwoord: je kan RT opslaan en de cycles toch runnen. Je kann ook naar de volgende trial. Maakt niet heel veel uit

Next up:

Alle data verzamelen en plotten?

Aparte script maken, aan Noor vragen voor haar Jupyter notebook?

Design vraag voor mergen: Wil je meerdere tasks willen runnen achter elkaar? Want dan moet je parameters dus meerdere malen importeren.

Next task:

Design:

* Parameters:
  + 2 opties:
  + Of je runt 1 experiment tegelijk, dan hoef je maar 1 keer parameters\_exp callen.
  + Vervolgens,:
    - Of, in elke sub script call je de juiste parameters op basis van een pickle? Niet efficient
    - Of, je maakt parameters task-dependent:
      * Get\_params(task): if experiment, haal dezelfde op
    - importeert alle parameters matrix in main, en geeft alle relevante parameters door aan de desbetreffende functies.
    - Of je maakt van de paramaters een grote dataframe, geeft deze door aan alle subscripts.
* Main en main\_exp: Done
* Task attributes:
  + Apart task attributes file,
  + Declare all task attributes in main?
  + Separate class in script? Like task.ncycles

Huidige structuur:

Elke task heeft zijn eigen attributen, en vraagt om eigen parameters.

Dus:

In het begin van main exp wordt global\_params geimporteerd. Daar staat in welke task er gerunt moet worden.

Vervolgens wordt task\_attributes geroepen op basis van welke task het is: dat is een object met alle attributen van de task to be ran.

Bvb: task\_attributes.language =’french’ als de task Flanker is

Daarna wordt task\_params geroepen, op basis van de attributen. Deze is verschillend, afhankelijk van of de task een experiment is of niet.

Bvb. Task\_params.epsilon= 0.035

Vervolgens, ipv dat in elke sub-module parameters\_exp.py of parameters.py nogmaals importeert, wat voor conflicten zorgt, zijn parameters en attributes argumenten van de functie die de submodules roept.

Bvb: ipv van

Main.py:

Import parameters\_exp

Simulate\_experiments()

Simulate\_experiments.py:

Import parameters\_exp

->

Main.py:

Import parameters\_exp

Simulate\_experiments(parameters)

* Affixes lijst met frequency erbij,
* Activatie via bigrams zoals alle anderen

Simulate\_experiments:

Contatenation van individual\_words (stimulus van task) en freq\_pred\_dict om lexicon.dat te maken.

Maar freq\_pred\_dict zou alle stimuli al moeten bevatten.

Waarom deze samenvoeging?

* Antwoord: freq\_pred\_dict missen sommige woorden: woorden die niet in de SUBTLEX waren zijn er uit gehaald. Door individual\_words er weer bij te doen ix het weer complete.
* Maar de echte vraag is: waarom is er een threshold in create\_freq\_pred?

Take into account eccentricity and edge weight? NU:nee

Bigrams vervangen positie 1 en last? Interference met edge-highlight. Of handmatig speciale bigrams invoegen met positie None? Nu: apparte lijst special bigrams

Worden met dezelfde affix extra inhibite op elkaar? Dus de bigrams ook gebruiken om inhibite matrix te maken? Geen probleem

\_p and pe in peer hebben zelfde positie en dezeldfde edge weight.

Pickle inhibitie matrix

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

Maak je affixes een speciale klasse? I.e. er wordt geen inhibitie op berekend met betrekking tot andere woorden.

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 het erg dat er dus een specifiek onderscheid onstaat tussen affixes en woorden? Qua aannames

Question: Alle woorden worden gebruikt voor bouwen inhibite matrix, affix niet meenemen? Dus bvb affix ment\_, moet dat in competetie staan met het woord \_mint\_? Of is het een aparte categorie zonder inhibite waardes? OK

Comment: wanneer je inhibite onafhankelijk maakt van word length, herken je feitelijk niets meer, omdat alles inhibite op elkaar uitoefent. BVB ‘we’ inhibibeert ‘weak’, waardoor weak niet herkend wordt als het de target is.

Wat te doen, lengte-criterium terug brengen, of weglaten maar inhibite waardes tweaken?

DONE: word-to-word inhibitie van -0.07 naar -0.005 verschoven (was een van de vorige waardes)

Comment: In aparte gevallen is less zowel een affix als een echt woord, met verschillende freq waardes. Dus de affixes lijst bij de lexicon doen zorgt misschien voor problemen

* Een optie is om niet tijdens splitsen naar bigrams de \_ toe te voegen, maar daarvoor al. Dus de lexicon bevat dan al woorden zoals \_hello\_ and ity\_. Dan heb je \_less\_ en less\_. Dus dit impliceert dat het lexicon dat een persoon heeft onderscheid maakt tussen die twee versies van less.

Nu: affixes worden meegenomen in inhibite, inhibitie is weer afhankelijk van lengte en \_less\_ en less\_ bestaan beiden in het lexicon.

If ok, dan is het mechanisme van affixes en implementeren van underscores system klaar

Volgende stap:

Mechanisme bouwen dat affixes herkent en een extra slot openmaakt voor de stam.

Affixes veranderen naar suffixes. DONE: affixes = suffixes + prefixes, waarbij prefixes nu leeg is

Logging module gebouwd

Idee voor pickle inhibitie 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

* Je krijgt dus niet voor elke matrix in het verleden een aparte file, maar alleen de meest recente.

Test : testen met non-words +suffix: milken

Bevindingen: hangt heel erg af van inhibitie waardes. Bij hoge waardes worden lange woorden nooit herkend. Bvb. Importantness.

Absolutely word herkend, maar niet als de stimulus absolutelyness is

Zowel absolutely als ness als absolutelyness wordt herkend, maar dan wel met hele lage inhibite waardes: 0.0005 (ipv 0.0165).

Y in absolutely wordt pas later herkend.

Interestingly: interesting wordt pas herkend als inhibite echt heel laag staat. Ly\_ wordt niet herkend.

NOTE: wanneer een woord niet in de 200 most frequent is, dan krijgt hij een freq van 0. Hoe beinvloedt dit de herkenning?

TODO: How to stabilize inhibition waardes? Via optimize functies van de code?

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

Beyersman 2: verschil in affix herkenning tussen duits en frans.

TODO: affix frequency lijst vinden, voor frans en duits.

Check inhibitie matrix: Alles ok

Alle negatieve waardes van inhibitie matrix?

Theorie: CORNER en HIGHER primen beide hun stem. Maar, CASHEW primet CASH niet! Dus alhoewel CASHEW de bigrams van CASH activeren, zorgt de inter-word inhibite ervoor dat CASH dan weer precies zoveel geinhibeerd wordt.

Maar hoe leg je dan de priming van CORNER EN HIGHER uit? Er is iets van een higher order process dat discrimineert tussen real-suffix en non-suffix, dat de stem primet. Hoe kan het system ingebouwd in OB1 wel voor priming daar zorgen ? Woorden zijn nog niet gesplit tijdens prime, dus die extra module heeft geen effect.

Aangezien er in het geval van CORNER en HIGHER splitting plaatsvindt van stem en suffix, zoek je specifiek voor het woord

-literatuur toont aan dat woord herkenning zo ongeveer 500ms duurt. Dus niet per se te snel, misschien juist te snel?

-max bigram verschil aanpassen

max\_threshold kann ook naar beneden

\_Inhibitie resultaten: meest robuust bij een middelhoge activatie en lage inhibitie. Bij higere excitatie, te veel random woorden. Bij hogere inhibitie, geen lange woorden meer herkend. Als je de balans in stand houdt kann je zowel exciatie en inhibitie naar boven doen. Dezelfde woorden worden dan herkend, maar sneller. Bij fixe inhibitie en excitatie heeft bigram-to-word inhibite als effect dat minder random woorden boven de threshold.

* Stabiele resultaten bij excitatie = 2.8, word2word\_inhib= 0.0006, bigram2word\_inhib= -0.14

Bij excitatie = 6, bigram2word\_inhib= -0.05 en word2word\_inhib= 0.008, worden affixes vrij consistent in de eerste 2 cycli herkend. Maar, is wel vrij instabiel: absolutelyness en importantness worden net wel of net niet herkend (nooit, zelfs niet met 30 seconden stimulus). , al is het woord toegevoegd bij de lexicon. Daarbij worden heel veel andere woorden ook actief.

(importantness en absolutelyness hadden eerst een freq van 0, nu minimum van dict).

Notebook werkt weer!

Comment: de prime wordt nu direct herkend, dus reactie word gelijk vastgelegd en alle trials zijn fout. Dus je wilt eigenlijk een mechanisme waar als eerste alleen de affix wordt herkend -> zoekt voor korter woord, en daarna wordt het woord herkend. Dus je moet een manier vinden om affixen veel herkenbaarder te maken dan lange woorden.

1. Affixen moeten geen word-to-word inhibition hebben. word-to-word op 0 voor alle woorden.

* Alleen 0 inhib op elkaar, of op alles, of alleen op woorden? Dus kan \_less\_ less\_ inhiberen en vice versa?
  + Nu: affixes hebben geen inhibite op woorden en vice versa. Maar affix wel op affix, maar woord wel op woord
* Easiest way to do this is change overlap between affixes

1. Threshold voor affixen lager zetten: frequency voor affixes verdubbelen.

Comment: vervolgens komen er veel affixes naar boven, zelfs age\_ voor tighten.

Met lage threshold + geen inhibitie: voorkeur voor korte woorden.

IDEE: inhibitie weegt zwaarder op lange woorden, omdat je meer overlap hebt. Dit zou je kunnen normailzeren via door de totale hoeveelheid bigrams+monograms te delen.

* Checken of dit niet al gebeurt.
* Nee, maar excitatie is eigenlijk al afhankelijk van lengte. Dus als inhibitie afhanklijk is van lengte, zou het zich ongeveer moeten balanceren. Maar daar heb je weinig controle over. Je kan dus excitatie normalizeren, en inhibite nornmalizeren: lange woorden zullen dan geen problemen meer opleveren.
* Of inhibitie wegen met activatie van woorden?

Vraag: threshold functie: zie 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.

Vraag: cumulative activation function?

Threshold mag veranderd worden: tot 0.6

Thresholds van affixes zijn bijzonder laag: naar kijken

Bigram gap veranderen: lager -> hoger 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

Mail begint fgb.

Martijn in cc

Dubbel frequency weghalen

We moeten een goede manier vinden zodat CORNER CORN activeert.

1. Wanneer woord in CORN + ER gesplitst wordt, CORN boost in activatie op woord level?
   1. Je krijgt dan het probleem dat WRIT+ER het woord WRIT zal activeren, en dat bestaat niet
2. Op herkenning van affix, threshold lager maken voor lengtes van stimulus-affix. Voor x cycli.
3. Alternatief: attention meer richten naar de kant tegenover affix? Dan krijg je niet het effect dat alle non-gerelateerd 4 letter woorden ook actief worden.
4. Alternatief 2: bigrams specifiek van de stem activeren. Dus alle bigrams zonder de letters van de stem opnieuw activeren? (aka. Wanneer woord in CORN + ER gesplitst wordt, stimulus van CORN en ER dubbel zo salient? Dus,excitatie specifiek verhogen voor die bigrams? -> CORN activatie op bigram level)
5. Wanneer woord in CORN + ER gesplitst wordt, geen inhibite meer van CORNER op CORN? Of, inhibite van het woord CORNER op 0 voor alles?
6. «**Morphological decomposition based on the analysis of orthography”:** Bij elke affix hoor een groep stems die vaak ermee voorkomen. Het herkennen van een affix, primet al die woorden (misschien overlap met actieve bigrams) -> zorgt voor extra activatie van die woorden. Wel een extra (grote) aanname
   1. Nee dit zal niet werken.
7. Affix herkennen maakt de affix ‘invisible’ -> prime en target komen 100% overeen?
8. Inhibitie verlagen tussen CORNER en CORN (sinds het begin). Dus, alle woorden met affixes hebben geen inhibite op hun stem.
   1. Maar, hoe zit het dan met affixed woorden die niet precies een stem hebben (RELATE-RELATABLE): als je RELAT zoekt, vind je niets.
   2. Woorden die je niet kent? MILKEN-MILK. Of zelfs SARB-SARBY. Verwacht je priming? Inhibitie is geleerd, maar is dat dan realistich bij een woord dat je niet herkent?

2 Theorien: 1. Er bestaat een mechanisme dat autmotaisch affix herkent en parst -> in dit geval verwacht je ook verschil in woorden die je niet kent. MILKEN zal meer primen dan MILKEW. 2. Nee, je leert handmatig tussen elk bekend woord en zijn stem, dat die niet met elkaar in competitite moeten staan. In dat geval zal CORNER primen, maar MILKEN niet.

<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)

Als je beslist de inhibitie matrix 0 te maken voor woorden zoals CORNER-CORN , zal het model werken, maar hoe interpreteren we dit? Je leert voor elk paar niet de stem te inhiberen, zelfs al heeft de stem er niets mee te maken? Er gebeurt dan niets «mechanisch» meer dat voor de priming effecten zorgt, maar alles is afhankelijk van de matrix, die we met de hand bouwen.

Non-primed and primed reactie tijden dichter bij elkaar: je wilt dat de priming van related prime gecancelled wordt door word2word inhibition, en dan weer wordt gered door special affix system: zie punt 4 in rood hierboven.

Het is toch zeer handig als we de tijdspan korter maken:

* In meer dan 2 cycli is het makkelijker om complexe effecten te modelleren
* Op dit moment is het vaak cyclus 2: affix herkennen, cyclus 3: woord herkennen. Weining resolutie in score verschillen.
  + Maakt niet uit, want hoe we het affix system nu designen impliceert dat affix alleen herkend hoedt te worden, en dan worden alle woorden lagere threshold. Dus dat effect zal plaatsvinden, ongeacht of de prime nog in beeld is of niet.
* Opmerking: het hele “zoek woorden met length-length(affix)” is niet meer nodig.. Als affix herkennen het lageren van de threshold van de restante woorden instand brengt is dat genoeg.
* Opnerking: non-suffixed + suffixed zijn nu gelijk qua RT. Maar, suffix wordt vaak herkend ipv woord, omdat lage threshold. BVB: ‘tion’ in ‘edition’, vervolgens is stimulus ‘edit’. ‘tion’ blijft hangen en is hoogst geactiveerde woord. Of zelfs, ‘ant\_’ ipv ‘cat’.
  + Als je affixes threshold hoger maakt, heb je weer het problem dat ze niet in de eerste rondes worden herkend.
  + Simpelweg «if word not in affixes”?
* In eerste instantie: effect van verlagen threshold, zorgt ervoor dat heeeel veel woorden boven de threshold komen. Helaas word dan meestal niet de goeie het meest actieve woord.
* TODO: plot, idee krijgen van balans word2word inhib en bigram2word inhib.
* Oscillation in inhibitie waardes? Yes, inhibite waardes zijn afhankelijk van de activatie van de ronde ervoor: eerst hoge activ. -> hoge inhib -> lage activ. -> lage inhib etc.
* Huidig probleem: korte woorden hebben te lage threshold en te hoge activatie vergeleken met de rest.

Comment: met onze simulatie zou je verwachten dat «grassy» , «bloem» primet, omdat het herkennen van een affix alle 5 letter woorden primet. Is dat realistisch?

Comment: inhibitie matrix is in de toekomst gedefineerd door het model van hoe kinderen leren lezen. Dus het actuele model is nog niet gebaseerd op een 100% onveranderbare theorie. We kunnen dus zonder te veel aannames deze inhibite matrix veranderen, als er maar een goeie rationale achter staat, wat consistent blijft met ons idee van hoe deze matrix onstaat.

Comment: onze herkenning van affixes is onafhankelijk van zijn stem, maar klopt dat? Zou je segmentatie verwachten op SMARTPHONE? primet SMARTPHONE SMART of PHONE?

??? probleem met stimuli??? Speaker staat onder non-suffixed..

Antwoord: ja, maar onder unrelated: dus het maakt niet uit welke prime, is toch geen overlap

!lastig: play wordt gezien als pla + affix -y. Action wordt gezien als ac + tion. Ac heeft een edit distance van 2 met heel veel woorden.

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

! elk woord in matrix had tot nu toe altijd inhibite op zichelf: WEAK inhibeert WEAK. Er uit halen?

Volgende keer: naar TODO’s kijken voor verbeteringen, analyse resultaten affix\_system aan/uit

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

+ spacy of nltk, in staat om woorden te splitten op affix + stem. Gebruiken in plaats van LCS algorithme?

Frequency:

HAL is text corpus

MorpholexFR – Morpholex equivalent

Morpholex haalt frequency uit English Lexicon project

!!! definitie Zipf:

it equals log10(frequency per billion words). Affixes.py was dus niet accuraat.

SUBTLEX-UK is Zipf

English Lexicon project is gebaseerd op HAL frequency. Maar, dat is dus een absolute waarde, niet relatief tot corpus grootte.

French lexicon project neemt waardes over van Lexique383, die zijn in het frequency per million (of log10 ervan).

DUS: eerst, MorphoLex-EN van HAL naar Zipf converten, vergelijken met huidige affix waardes.

Dan Morpholex-FR naar Zipf.

Probleem, niemand weet hoe groot precies HAL is. Zo ongeveer 130-400 miljoen.

Probleem: Morpholex-EN zipf converted matcht niet met huidige affix waardes: zit er telkens overheen.

Voor volgende keer:

MorpholexFR en MorpholexEN worden gebruikt om affix frequency te meten. Maar! Scales zijn niet hetzelfde. Morpholex EN is gebaseerd op HAL (ruw woord aantal – convert to zipf), en FR volgens mij niet! Figure out welke schaal, en dan naar Zipf converteren. Engelse woord dict (gebaseerd op SUBTLEX-UK) is ook in zipf, check of SUBTLEX FR ook in zipf is. (niet dus)

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.

Idee: cross reference THE in andere databases om HAL grootte te meten

Ok! Dus: Morpholex FR neemt freq waardes aan van FLP, gevonden op Lexique.org. FLP neemt deze waardes over van Lexique383. Deze zijn in frequency per million.

HAL waardes vergeleken met grote online woord database (EN) (1 miljard)

* HAL is geschat op 400 miljoen

Tests: SUBTLEX UK en Morpholex waardes voor “hi” zijn hetzelfde in Zipf (als we uitgaan van een HAL grootte van ong 400 000 000): ongeveer 5.3. In de final freq\_pred files zijn deze waardes ook gelijk. Dus de affix freq waardes in engels kloppen .

Voor freq waardes moet wel duidelijk wat we precies willen gebruiken: frequency van de lemma, of van het specifieke woord? Op dit moment is het voor engels het woord zelf, maar SUBTLEX bevat ook waardes voor lemma frequency.

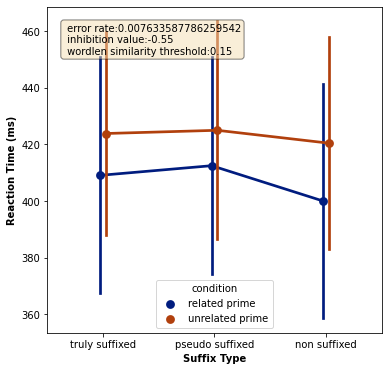
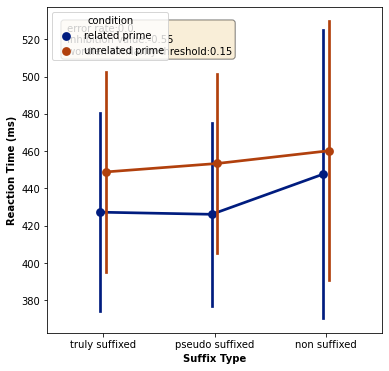
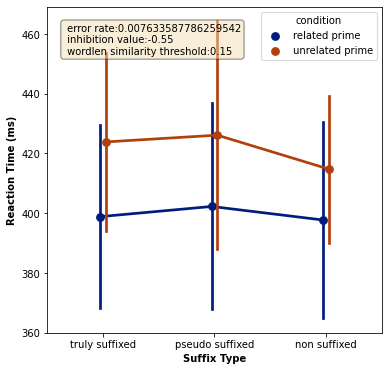
Lexique.org heeft frequency waardes in fpm, en deze waardes komen 1op1 overeen met de MorphoLex waardes (en ook met French Lexicon Project waardes). De huidige pred\_freq frequencies komen ook overeen met Morpholex. Idem als voor engels, de huidige woord frequencies zijn van het woord zelf en niet van de lemma.

TODO: uitleg toevoegen in README. Cleanup files

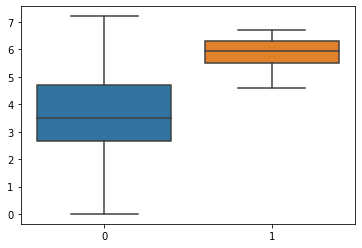
Update: Je verwacht geen priming van CASHEW op cash, maar we hadden dat wel. Inhibitie omhoog werkte niet!! Met meer inhibitie, alles geen priming. Gek, want inhibite van CORNER op CORN staat altijd op 0 (double check). Hoe kan dat? Dus GIRLDE primet CORN netzoveel als CORNER

Lag aan affixes: de waardes daarvan liggen nu hoger (van 6 naar 7.21) (maar, max is 7.67: \_the\_).

Huidig probleem: met het veranderen van de affix waardes verdwijnt het no-priming effect van non-suffixed:



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

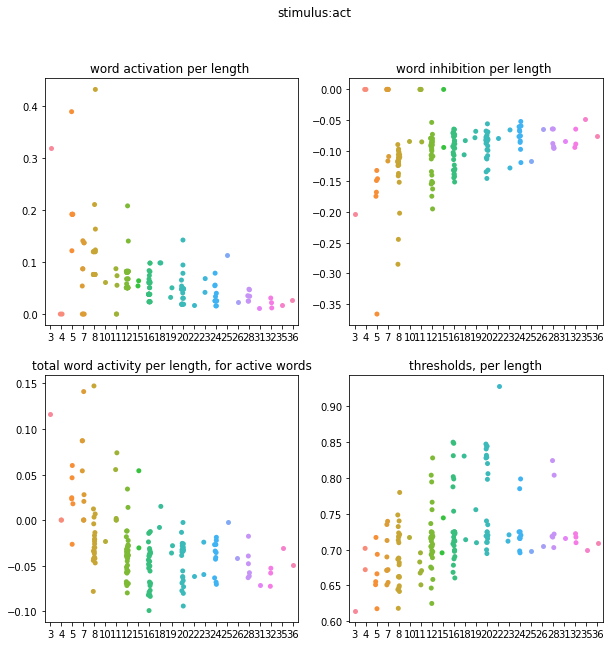
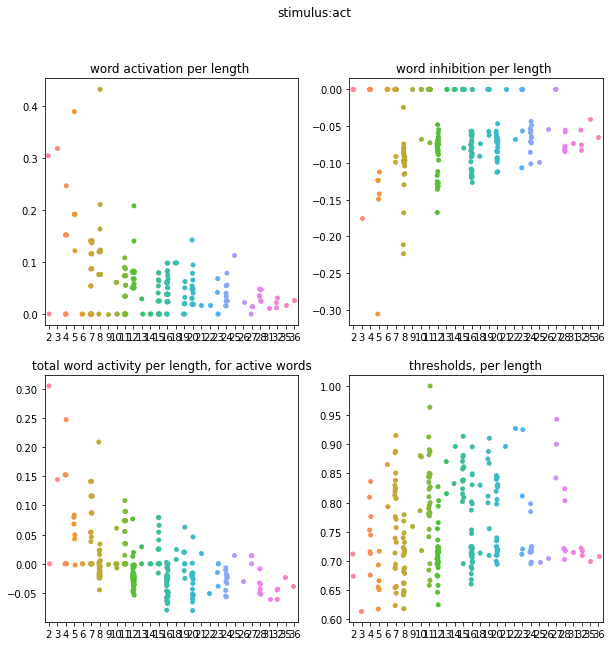


Affix frequency distribution

Oude waardes (oranje) zijn gemiddeld hoger, maar maximum is lager.

Wanneer de nieuwe waardes gebruikt worden, wordt in 1 geval fluffy te snel herkend terwijl het de prime is.

Nieuwe waardes zorgen voor hogere activatie?



New affix values old affix values

Comment: lexicon\_normalized\_word\_inhibition = (

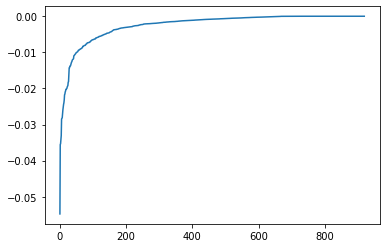
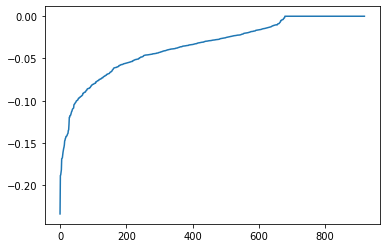
100.0/LEXICON\_SIZE) \* pm.word\_inhibition

Inhibitie wordt gedeeld door totale lexicon. Lexicon is nu veel groter.

Volgende stap: alleen suffixes houden die in Beyersmann voorkomen.

Vraag: inhibitie van alle affixes is 0.0. is dat realsitisch? Geen inhibitie tussen affixes?

Idee: Dat weaken, weak niet inhibeert, maakt een te klein verschil. Op dit moment is het zo dat als 100 woorden actief zijn, ongeveer alle woorden in het lexicon geinhibeerd worden. (100 woorden hebben minstens 1 letter overlap met alles). Deze inhibite meer gericht maken? Bigram gap verkleinen? Of van de inhibitie functie een exponentiaal maken? Dat alleen hoge overlap + activatie waardes echt inhiberen?



Normale curve -(kwadraat)

Werkt niet goed: bias towards korte woorden. (deze hebben minder overlap met anderen, dus

Merge Beatriz:

Wordfreq van NL is toegevoegd. Wat is daar de bron van?

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.

En hoe is de grammar predicatability van EN en FR toegevoegd? Zat POS in FLP en daaruit kan je predictability afleiden?

Nu heb je French\_Lexicon\_Project.txt (oud) en french\_lexicon\_tagged.csv (nieuw) en french\_lexicon\_tagged2000.csv (nieuw). Freq is gemiddeld van boeken en films? Zelfde waardes, dus txt vervangen met nieuwe CSVs? En scripts aanpassen.

Stimuli met POS: zijn dat stimuli of texts met informatie over 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: juiste database kiezen en POS integreren.

Simtae\_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

Systeem voor slots matching geimplementeerd? Zo ja, interactie met affix systeem (nieuw slot maken voor herkende stem)

Hoe zijn de POS bef, ProbBef etc. gemaakt? Bestonden deze files al via een ander experiment, of met de hand geschreven?

Subtlex NL: CD is number of documents the word appears in. Max 4, dus niet dezelfde measure als voor andere talen!

Maar vooral: POS prob matrix snap ik (is algemeen gedefineerd per taal), Stimuli met POS erbij ook (simpelweg Subtlex woorden met hun POS matchen),

maar hoe wordt de task POS file gemaakt? (zowel POS\_prob\_{task} in Texts als Sentence\_{task}\_POS in Stimuli)

Get\_pred\_freq kan naar create\_...

Vraag: Subtlex NL bevat al POS, waarom TreeTagger gebruikt? En waarom is er dominant.POS en POS?

Normalize\_pred\_values in reading\_functions?

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

Consistency checks in beginning

Get\_threshold: Comment zegt “compute thresh based on freq and pred”, maar pred is niet als input meegegeven?

Waarom pred values van word zelf, word erna en word ervoor?

Pred\_values is ambigu: staat in de code vermeldt voor elke task, en zou volgens de code gebruikt moeten worden voor de bepaling van de threshold. Alleen wordt pred\_value niet gebruikt door de threshold functie. De nieuwe taken, die wel pred\_values gebruiken, hebben een hele andere structuur, en gebruiken de threshold functie uberhaput niet. Dus hoe zit het met die pred values? Uit de code halen voor non-POS taken? i.e. niet medebepalend voor threshold? (is het nu toch niet).

Interactie slot-system en affix: wat moet er gebeuren als een affix is herkend? Moeten alle slots ook een woord van (woord – affix) accepteren (slots 4,5,6 accepteren ook 2,3,4)?

Dus: affix herkend. Je test eerst slot 0. Daar zoek je een woord van de lengte van de stimulus. Maar, datzelfde slot kan ook een woord matchen van de stem lengte.

Of misschien helemaal geen interactie? Op dit moment willen we alleen dat weaken weak primet, maar dat is gergeld via het inhibitie systeem. Priming vindt dus onafhankelijk van herkenning plaats. Dus is het nodig om de stem op een slot te kunnen matchen?

Je wilt in ieder geval niet dat door dit mechanisme, WEAK herkent wordt alsof het fitting length met WEAKEN is. Er bestaat priming, maar geen substitutie.

Op dit moment zet de code de activatie van een woord op 0 als deze herkend en gematcht is. Dit zorgt ervoor dat een verkeerd gematcht woord uit de weg gaat en het goeie woord een kans krijgt. Maar je kan het dus nooit echt fout doen, als het niet het goeie woord is ga je door tot het goeie gematcht is. Klopt dat?

Ook zorgt het ervoor dat het woord direct erna verdwijnt uit activiteit. Bij priming kan dat een probleem zijn, want als je activiteit op 0 zet zal je priming belemmeren.

En als je 3 keer hetzelfde hebt? WEAK WEAK WEAK? WEAK op slot zetten maar activatie houden?

Misschien cooldown periode van 2 cycli waar hij wel actief is maar niet gematcht kan worden?

Ook is het zo dat met het slot mechanisme je exponentieel veel fouten maakt hoe langer de zin is. Heel vaak woorden verkeerd gematcht. IE. My name is quintijn -> quit wordt gematcht op 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 nu in de cycli loop, want deze moet gerest worden wanner de stimulus verandert.

In general: POS en pred\_values lopen door elkaar: moeten duidelijker in de code gebruikt worden, maar daarvoor moet ik eerst antwoord krijgen van Beatriz.

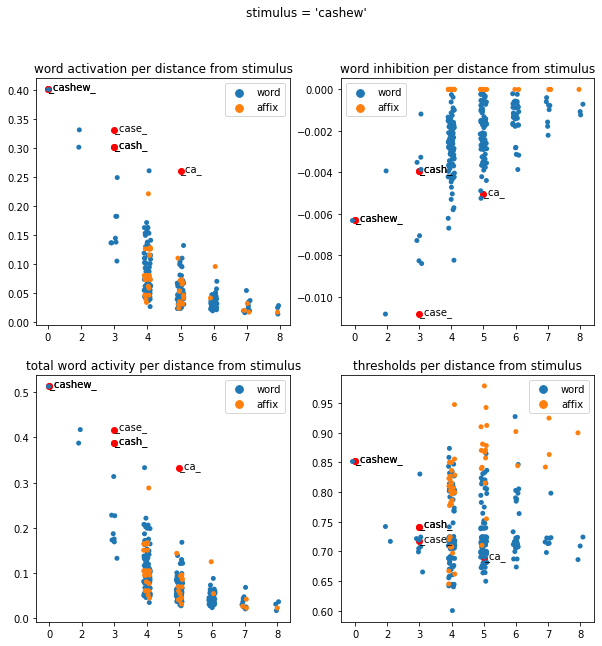
Inhibitie waardes probleem:

Recap:

Je verwacht dat met het affix systeem ( affixed woorden en hun stem hebben geen inhibitie op elkaar) dat affixed woorden gevolgd door hun stem eerder herkend zullen worden (priming effect) maar niet woorden die niet geaffixt zijn.

Echter gebeurt dit niet: er vind nauwelijks priming plaats.

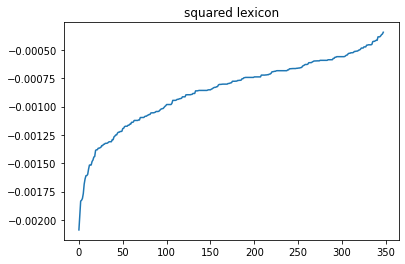
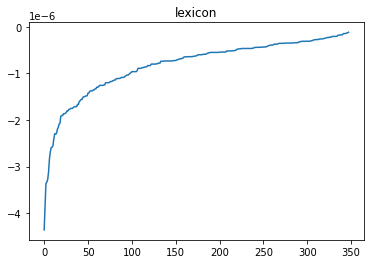
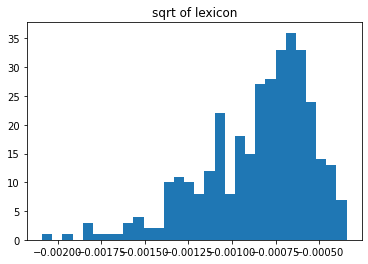
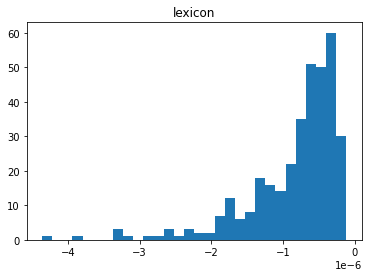
Test: activatie hoger met similarity bigrams

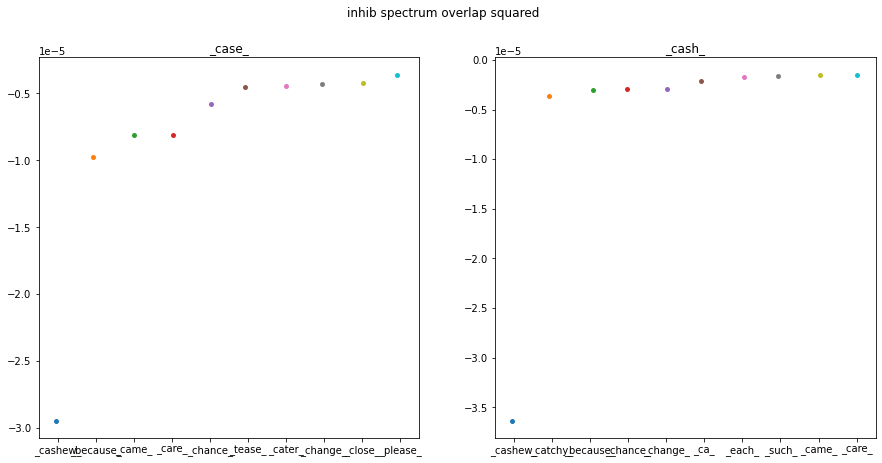
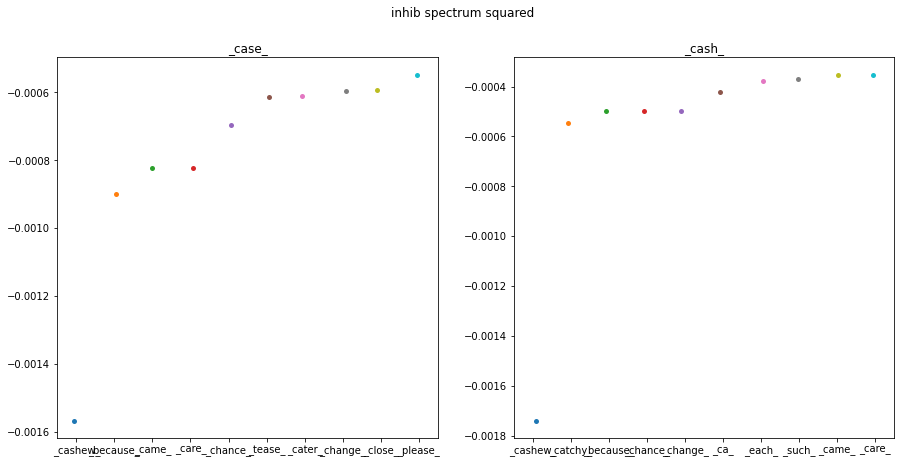
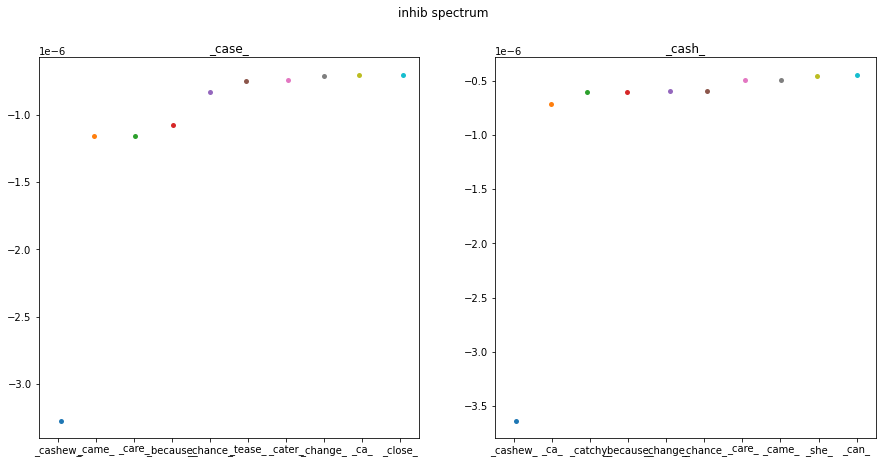


1. Activatie / inhibitie neemt af met distance
2. Cash heeft alsnog een hoge activatie, alhoewel we expiciet cashew Cash heel erg wouden laten inhiberen
3. Waarom heeft Case veel meer inhibitie als Cash?

Todo: plot voor cash and case welke woorden het meest verantwoordelijk zijn voor inhibititie

Lexicon squared:

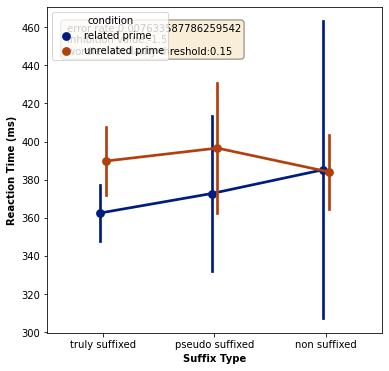
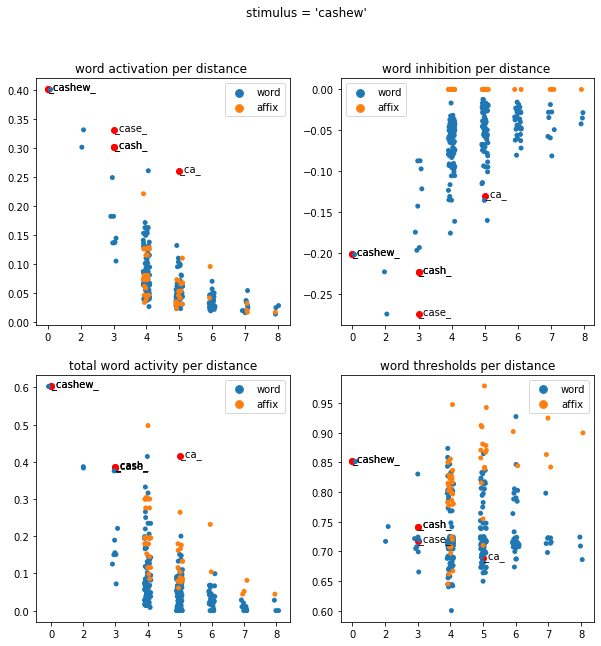


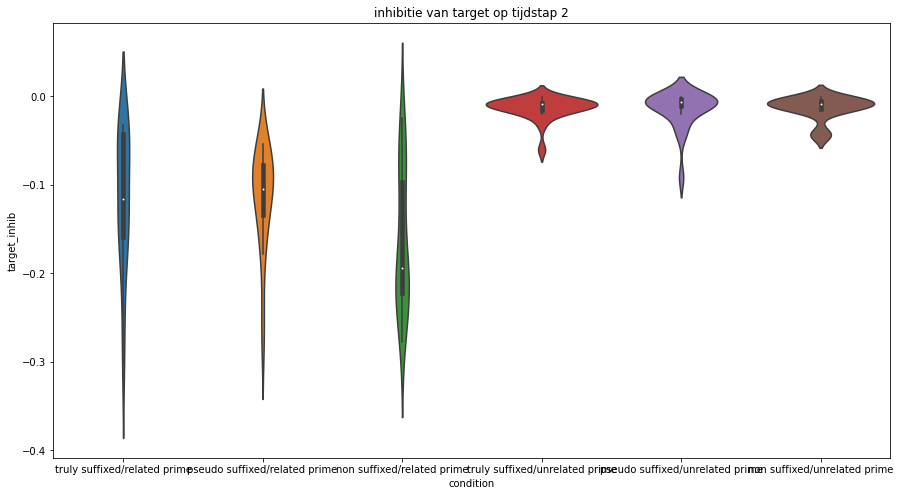


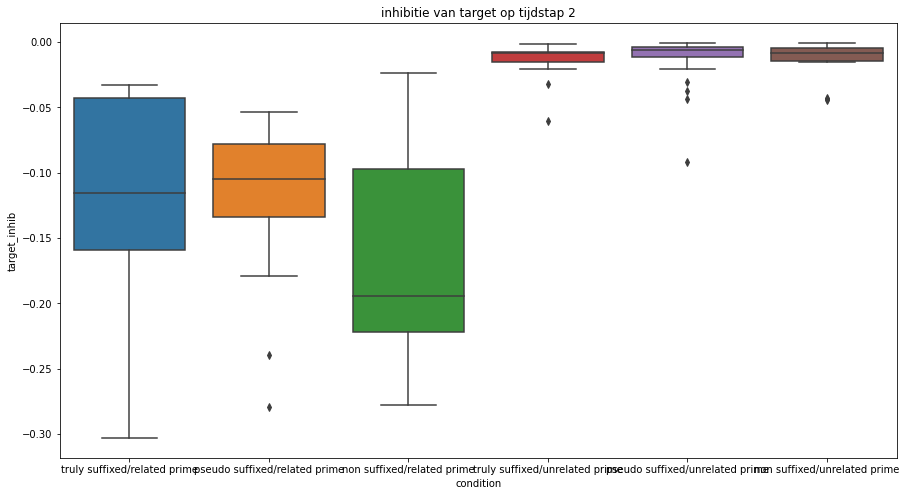
Next up: kijken naar squared + inhibitie cutoff

Results:

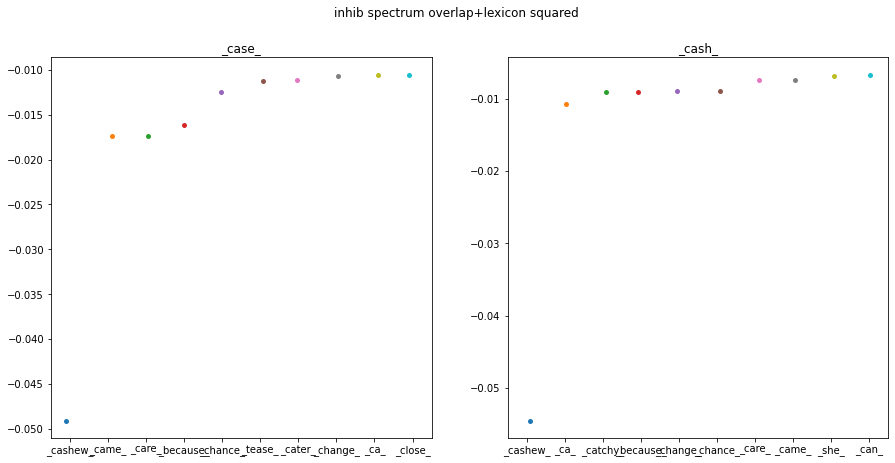
Inhibitie squared:

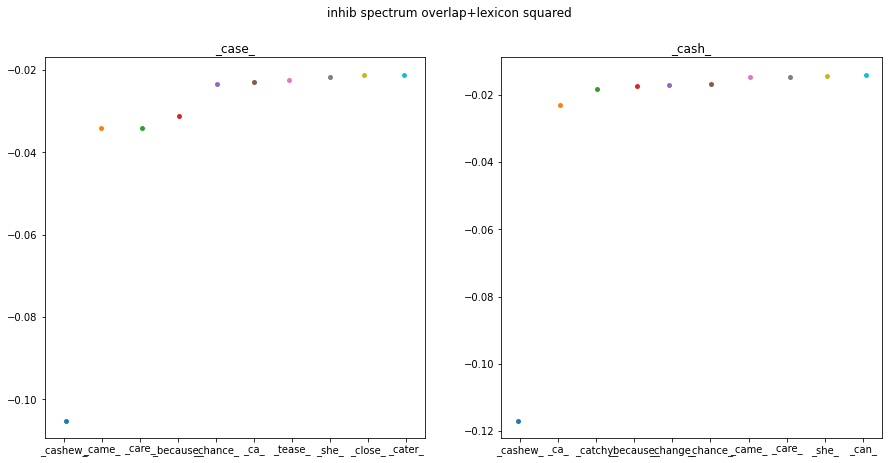






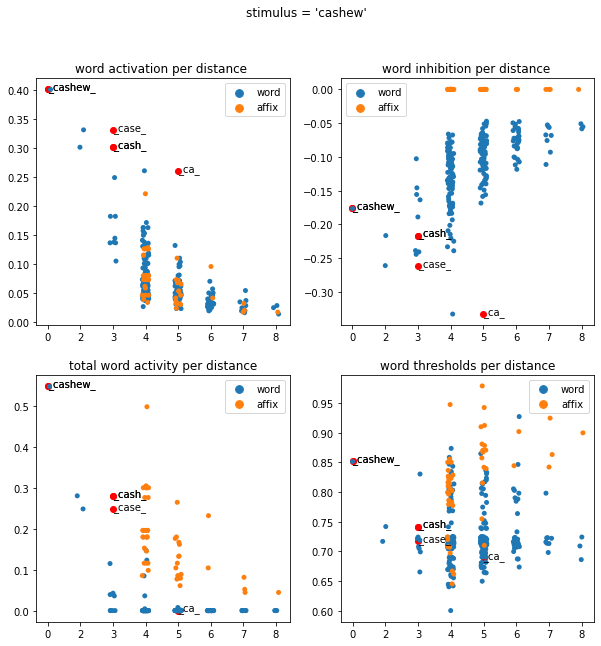
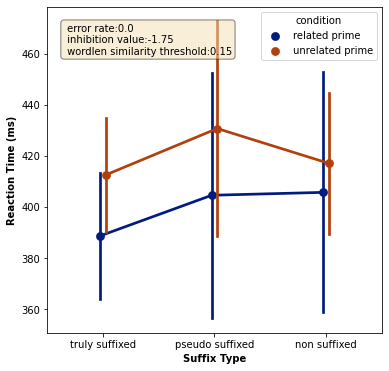
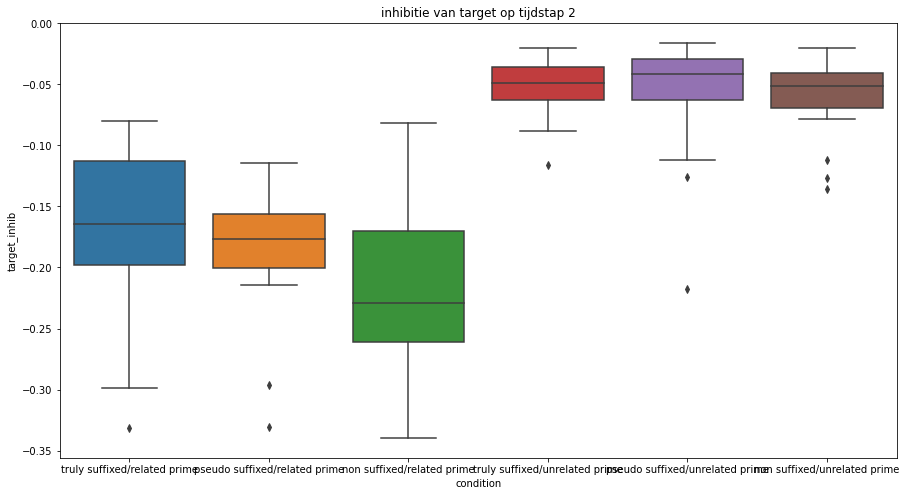
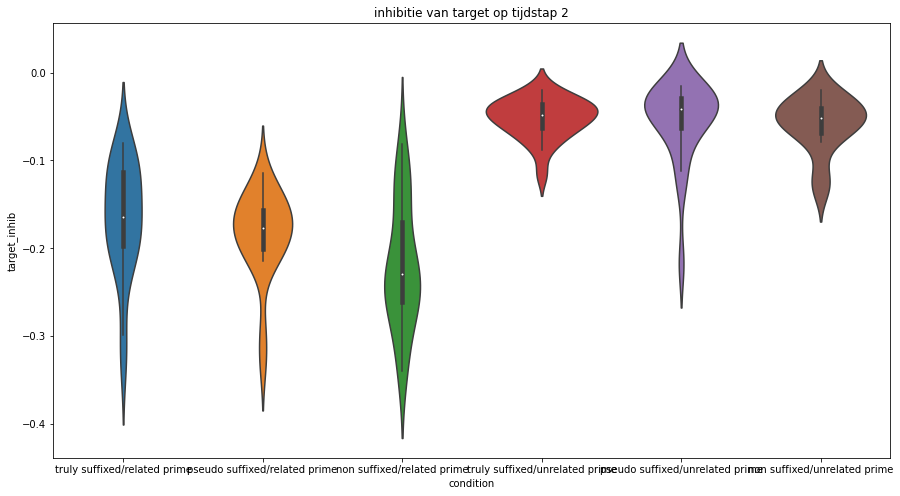
Tijdstap 1

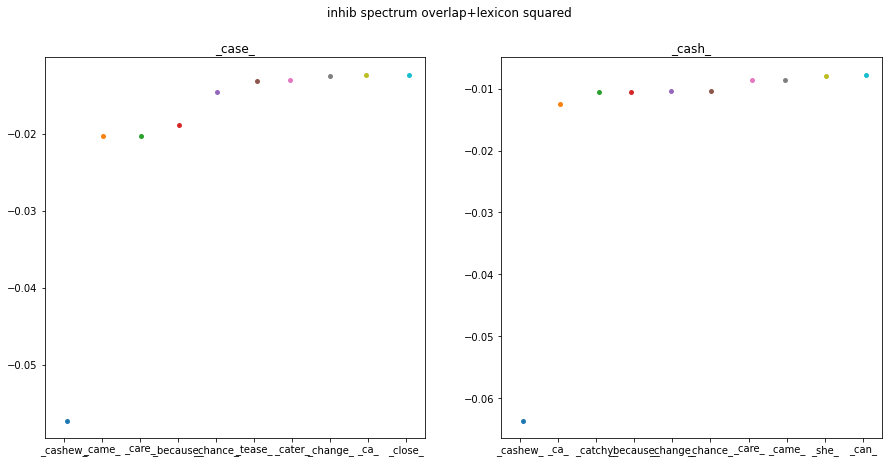


Tijdstap 2

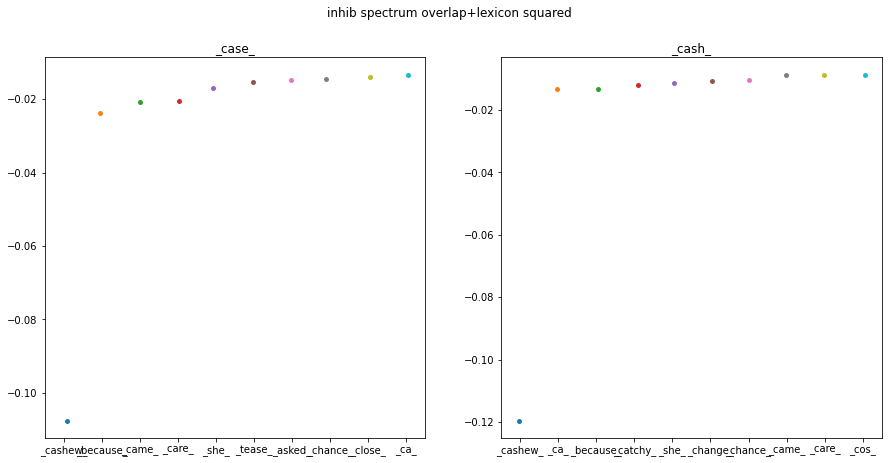
Note: bij hogere inhibitie waardes, specifiek voor related prime / non suffixed worden sommige woorden nooit herkend.

Inhibitie cutoff:



Tijdstap 1

Tijdstap 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 ipv. 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.