Neural Language Modeling Research

Author: Mijin Gwak

Overview

This repository contains the research project conducted during my research master's, focusing on investigating whether neural language models exhibit aspects of human bilingual sentence processing. The work, based on the insights presented by Frank (2021), involves the training and testing of bilingual Long Short-Term Memory (LSTM) language models. Unfortunately, the code for the training process couldn't be included due to copyright restrictions. However, I have uploaded the process of calculating and plotting the garden-path effects from the experiments.

Research Description

In this research project:

Objective: Explore the aspects of human bilingual sentence processing in neural language models.

- **Methodology:** Bilingual LSTM language models were trained on a substantial dataset comprising nearly 17 million Dutch and English sentences.
- **Testing:** Models were tested on sentences with local structural ambiguity (garden-path sentences) derived from human reading-time experiments.
- **Measurement:** The simulated garden-path effect was measured by comparing surprisal values of the critical (disambiguating) word between ambiguous and unambiguous versions of test sentences.
- **Findings:**
  - The garden-path effect was observed, primarily induced by the NP/S-coordination ambiguity.
  - Stronger garden-path effects were noted after priming with an unambiguous structure.
  - Within-language priming proved more effective than between-language priming.

  - Priming had a more pronounced effect on garden-pathing with a favorable thematic condition.
  - Additional experiments, reducing the training corpus for one language by half, yielded consistent results.
  - The garden-path effect was analyzed concerning target sentence proficiency, revealing stronger effects with high proficiency (L1) than low proficiency (L2).
- **Conclusion:** The results suggest that LSTMs exhibit human-like structural adaptation within and across languages.

Key Concepts

NP/S-coordination Ambiguity The NP/S-coordination ambiguity refers to a local syntactic ambiguity in a sentence where a noun phrase (NP) could be interpreted either as part of an object NP conjunction or as the beginning of a new clause. In the context of this research, the ambiguity arises in sentences such as:

(1a) The wizard guards the king and the princess protects the prince with her life.
(1b) The wizard guards the king, and the princess protects the prince with her life.

The disambiguation occurs when the critical verb ("protects") is encountered, leading to increased reading time in ambiguous sentences compared to unambiguous variants.

Local Structural Ambiguity Local structural ambiguity, often referred to as garden-path sentences, occurs when a sentence contains elements that mislead the reader initially, causing temporary confusion or a "garden-path" effect. In the context of this research, the garden-path sentences involve NP/S-coordination ambiguity, and the models were tested on their ability to navigate and adapt to such local structural ambiguities.

Additional Experiment

An extra experiment involved reducing the training corpus for one language by half: - Specifically, 50% of the English training corpus was removed to elicit an L1 Dutch - L2 English model. - Similarly, 50% of the Dutch training corpus was removed to elicit an L1 Dutch - L2 English model. - Results from this experiment were consistent with the findings of the initial experiment.

Reference

- Frank, S.L. (2021), "Towards computational models of multilingual sentence processing," Language Learning 71, pp. 193–218.

I Preliminaries

1. R Packages Used

```
library(tidyverse) # general tools
library(ggplot2) # plots
library(DiagrammeR) # flowcharts
```

2. Types of Priming

There are two *prime types* and *target types*, respectively (i.e., ambiguous, unambiguous), two *prime languages* and *target languages*, respectively (i.e., Dutch, English) as below. This priming procedure was performed with two *thematic fit conditions* (i.e., good, bad), which elicits 32 conditions (2x2x2x2x2) in total.

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is i
```

3. Garden-Path Effect

Garden-path effect refers to the surprisal difference between the unambiguous and ambiguous targets of the same priming condition (i.e., nl_amb, nl_unamb, en_amb, en_unamb). Therefore, target type is removed from the condition which results in 8 conditions (2x2x2) per thematic condition, and 16 conditions in total:

II Prime Tests Results

1. First Experiment

In the first experiment, the Five Long short-term memory (Hochreiter & Schmidhuber, 1997) RNN models, differing only in random initial connection weights, were trained on next-word prediction for Dutch and English. The training corpus consisted of nearly 17 million sentences (225 million word tokens) scraped from web sources (Schäfer, 2015).

In order to analyze the garden-path effect, the surprisal differences *surpdiffs* will be categorized by *prime type* (i.e., ambiguous, unambiguous), and *language pair* (i.e., within-language, between-language) for each thematic fit condition:

The following function calculates surprisal differences between two prime types (i.e., ambiguous - unambiguous) and adds relevant labels (i.e., prime type, language pair, target sentence proficiency).

param *df*: dataframe of prime test results
param *prime_type*: prime type of the values
param *target_amb*: values of ambiguous target type which will subtract
param *target_unamb*: values of unambiguous target type
return: a dataframe with surprisal differences, prime type and language pair

```
calculate_surpdiffs_1 <- function(df, prime_type, target_amb, target_unamb){

  # select values with which to calculate the differences in surprisal

  df_target_amb <- df[df$prime_type == prime_type &
                        df$target_type == target_amb &
                        df$prime_nr != df$target_nr,] # exclude identity priming

  df_target_unamb <- df[df$prime_type == prime_type &
                          df$target_type == target_unamb &
                          df$prime_nr != df$target_nr,]

  # calculate surprisal differences

  surpdiffs <- as.data.frame(df_target_amb$surp - df_target_unamb$surp) %>%
    setNames('surpdiffs')

  # Create a new column for prime type
  # Assign 'Ambiguous' when the prime type is either 'amb_nl' or 'amb_en'
  # Assign 'Ambiguous' when the prime type is either 'unamb_nl' or 'unamb_en'

  amb <- c('nl_amb', 'en_amb')
  unamb <- c('nl_unamb', 'en_unamb')

  surpdiffs$prime_type <- ifelse(is.element(prime_type, amb), 'Ambiguous', 'Unambiguous'

  # Create a new column for language pair
  # Assign 'within' when the first two letters of prime type (i.e., nl, en) are equal
  # Assign 'cross' when they are not equal

  surpdiffs$language_pair <- ifelse(substr(prime_type, 1, 2) == substr(target_amb, 1, 2)

  return(surpdiffs)
```

```
  }
```

1.1. *Good* Thematic Condition

Retrieving data:

```
exper1_good <- read.csv('LSTM_1_17750580_good.csv') %>%
  select(prime_type, target_type, prime_nr, target_nr, surp)
```

```
##   prime_type target_type prime_nr target_nr    surp
## 1   nl_unamb    nl_unamb        0         0  9.9216
## 2   nl_unamb    nl_unamb        0         1  9.8551
## 3   nl_unamb    nl_unamb        0         2  9.6665
## 4   nl_unamb    nl_unamb        0         3 12.1319
## 5   nl_unamb    nl_unamb        0         4 12.1645
## 6   nl_unamb    nl_unamb        0         5 12.4961
```

Calculating garden-path effect with *calculate_surpdiffs_1* function for each priming condition:

```
# Prime type: Ambiguous

exper1_good_1 <- calculate_surpdiffs_1(exper1_good, 'nl_amb', 'nl_amb', 'nl_unamb')
exper1_good_2 <- calculate_surpdiffs_1(exper1_good, 'nl_amb', 'en_amb', 'en_unamb')
exper1_good_3 <- calculate_surpdiffs_1(exper1_good, 'en_amb', 'nl_amb', 'nl_unamb')
exper1_good_4 <- calculate_surpdiffs_1(exper1_good, 'en_amb', 'en_amb', 'en_unamb')

# Prime type: Unambiguous

exper1_good_5 <- calculate_surpdiffs_1(exper1_good, 'nl_unamb', 'nl_amb', 'nl_unamb')
exper1_good_6 <- calculate_surpdiffs_1(exper1_good, 'nl_unamb', 'en_amb', 'en_unamb')
exper1_good_7 <- calculate_surpdiffs_1(exper1_good, 'en_unamb', 'nl_amb', 'nl_unamb')
exper1_good_8 <- calculate_surpdiffs_1(exper1_good, 'en_unamb', 'en_amb', 'en_unamb')
```

Combining the results from all the priming conditions:

```
exper1_good_results <- rbind(exper1_good_1, exper1_good_2, exper1_good_3, exper1_good_4,
                             exper1_good_5, exper1_good_6, exper1_good_7, exper1_good_8)
```

```
##   surpdiffs prime_type language_pair
## 1    0.0539  Ambiguous        Within
## 2    0.4773  Ambiguous        Within
## 3    0.5961  Ambiguous        Within
## 4   -0.0232  Ambiguous        Within
## 5    0.0054  Ambiguous        Within
## 6    2.6542  Ambiguous        Within
```

1.2. *Bad* Thematic Condition

Retrieving data:

```
exper1_bad <- read.csv('LSTM_1_17750580_bad.csv') %>%
  select(prime_type, target_type, prime_nr, target_nr, surp)
```

```
##   prime_type target_type prime_nr target_nr    surp
## 1   nl_unamb    nl_unamb        0         0  8.9860
## 2   nl_unamb    nl_unamb        0         1  8.8871
## 3   nl_unamb    nl_unamb        0         2 15.1430
## 4   nl_unamb    nl_unamb        0         3 12.4412
## 5   nl_unamb    nl_unamb        0         4 14.5780
## 6   nl_unamb    nl_unamb        0         5 10.5293
```

Calculating garden-path effect with *calculate_surpdiffs_1* function for each priming condition:

```
# Prime type: Ambiguous

exper1_bad_1 <- calculate_surpdiffs_1(exper1_bad, 'nl_amb', 'nl_amb', 'nl_unamb')
exper1_bad_2 <- calculate_surpdiffs_1(exper1_bad, 'nl_amb', 'en_amb', 'en_unamb')
exper1_bad_3 <- calculate_surpdiffs_1(exper1_bad, 'en_amb', 'nl_amb', 'nl_unamb')
exper1_bad_4 <- calculate_surpdiffs_1(exper1_bad, 'en_amb', 'en_amb', 'en_unamb')
```

```
# Prime type: Unambiguous

exper1_bad_5 <- calculate_surpdiffs_1(exper1_bad, 'nl_unamb', 'nl_amb', 'nl_unamb')
exper1_bad_6 <- calculate_surpdiffs_1(exper1_bad, 'nl_unamb', 'en_amb', 'en_unamb')
exper1_bad_7 <- calculate_surpdiffs_1(exper1_bad, 'en_unamb', 'nl_amb', 'nl_unamb')
exper1_bad_8 <- calculate_surpdiffs_1(exper1_bad, 'en_unamb', 'en_amb', 'en_unamb')
```

Combining the results from all the priming conditions:

```
exper1_bad_results <- rbind(exper1_bad_1, exper1_bad_2, exper1_bad_3, exper1_bad_4,
                            exper1_bad_5, exper1_bad_6, exper1_bad_7, exper1_bad_8)
```

```
##   surpdiffs prime_type language_pair
## 1    1.0433  Ambiguous        Within
## 2   -0.1828  Ambiguous        Within
## 3    1.0472  Ambiguous        Within
## 4    0.6238  Ambiguous        Within
## 5    0.9901  Ambiguous        Within
## 6    1.0622  Ambiguous        Within
```

2. Second Experiment

In the second experiment, the training corpus for *one language* was reduced by half resulting in about 13 million in total. Specifically, 50% of the English training corpus was removed to elicit an L1 Dutch - L2 English model, and 50% of the Dutch training corpus was removed to elicit an L1 Dutch - L2 English model. In both models, all four types of priming (i.e., nl_amb, nl_unamb, en_amb, en_unamb) were performed.

The garden-path effect in this experiment will be analyzed in the same way as the first experiment, but the target sentence proficiency (i.e., L1, L2) was also taken into account:

The prime tests were performed for both L1 Dutch - L2 English model and L1 Dutch - L2 English model independently, and therefore the results are documented in two separate files. The surprisal differences will be calculated separately, then labels will be added for prime type, language pair, and target sentence proficiency to each model. Finally, the results from the two models will be combined for the garden-path effect analysis.

The following function calculates surprisal differences between two prime types (i.e., ambiguous - unambiguous) and adds relevant labels (i.e., prime type, language pair, target

sentence proficiency):

param *df*: dataframe of prime test results
param *prime_type*: prime type of the values
param *target_amb*: values of ambiguous target type which will subtract
param *target_unamb*: values of unambiguous target type
param *L1*: L1 of the model for which full training corpus was used
return: a dataframe with surprisal differences, prime type, language pair, and target sentence proficiency

```
calculate_surpdiffs_2 <- function(df, prime_type, target_amb, target_unamb, L1){

  # select values with which to calculate the differences in surprisal

  df_target_amb <- df[df$prime_type == prime_type &
                        df$target_type == target_amb &
                        df$prime_nr != df$target_nr,] # exclude identity priming

  df_target_unamb <- df[df$prime_type == prime_type &
                          df$target_type == target_unamb &
                          df$prime_nr != df$target_nr,]

  # calculate surprisal differences

  surpdiffs <- as.data.frame(df_target_amb$surp - df_target_unamb$surp) %>%
    setNames('surpdiffs')

  # Create a new column for prime type
  # Assign 'Ambiguous' when the prime type is either 'amb_nl' or 'amb_en'
  # Assign 'Ambiguous' when the prime type is either 'unamb_nl' or 'unamb_en'

  amb <- c('nl_amb', 'en_amb')
  unamb <- c('nl_unamb', 'en_unamb')

  surpdiffs$prime_type <- ifelse(is.element(prime_type, amb), 'Ambiguous', 'Unambiguous'

  # Create a new column for language pair
  # Assign 'within' when the first two letters of prime type (i.e., nl, en) are equal
  # Assign 'cross' when they are not equal

  surpdiffs$language_pair <- ifelse(substr(prime_type, 1, 2) == substr(target_amb, 1, 2)

  # Create a new column for target sentence proficiency
  # Assign 'L1' when the L1 parameter equals to the first two letters of target type
```

```
  # Assign 'L2' when they are not equal

  surpdiffs$target_prof<- ifelse(L1 == substr(target_amb, 1, 2), 'L1', 'L2')

  return(surpdiffs)

  }
```

2.1. L1 Dutch - L2 English Model

2.1.1. *Good* Thematic Condition

Retrieving data:

```
nl_en_good <- read.csv('LSTM_1_13363796_nl_en_good.csv') %>%
  select(prime_type, target_type, prime_nr, target_nr, surp)
```

```
##   prime_type target_type prime_nr target_nr    surp
## 1   nl_unamb    nl_unamb        0         0 10.0873
## 2   nl_unamb    nl_unamb        0         1  9.8062
## 3   nl_unamb    nl_unamb        0         2  9.6775
## 4   nl_unamb    nl_unamb        0         3 11.6247
## 5   nl_unamb    nl_unamb        0         4 12.2725
## 6   nl_unamb    nl_unamb        0         5 12.5695
```

Calculating garden-path effect with *calculate_surpdiffs_2* function for each priming condition:

```
# Prime type: Ambiguous

nl_en_good_1 <- calculate_surpdiffs_2(nl_en_good, 'nl_amb', 'nl_amb', 'nl_unamb', 'nl')
nl_en_good_2 <- calculate_surpdiffs_2(nl_en_good, 'nl_amb', 'en_amb', 'en_unamb', 'nl')
nl_en_good_3 <- calculate_surpdiffs_2(nl_en_good, 'en_amb', 'nl_amb', 'nl_unamb', 'nl')
nl_en_good_4 <- calculate_surpdiffs_2(nl_en_good, 'en_amb', 'en_amb', 'en_unamb', 'nl')
```

```
# Prime type: Unambiguous

nl_en_good_5 <- calculate_surpdiffs_2(nl_en_good, 'nl_unamb', 'nl_amb', 'nl_unamb', 'nl'
nl_en_good_6 <- calculate_surpdiffs_2(nl_en_good, 'nl_unamb', 'en_amb', 'en_unamb', 'nl'
nl_en_good_7 <- calculate_surpdiffs_2(nl_en_good, 'en_unamb', 'nl_amb', 'nl_unamb', 'nl'
nl_en_good_8 <- calculate_surpdiffs_2(nl_en_good, 'en_unamb', 'en_amb', 'en_unamb', 'nl'
```

Combining the results from all the priming conditions:

```
nl_en_good_results <- rbind(nl_en_good_1, nl_en_good_2, nl_en_good_3, nl_en_good_4,
                            nl_en_good_5, nl_en_good_6, nl_en_good_7, nl_en_good_8)
```

```
##   surpdiffs prime_type language_pair target_prof
## 1    0.0060  Ambiguous        Within          L1
## 2   -0.2901  Ambiguous        Within          L1
## 3   -0.2153  Ambiguous        Within          L1
## 4    0.0465  Ambiguous        Within          L1
## 5   -0.2341  Ambiguous        Within          L1
## 6    0.4800  Ambiguous        Within          L1
```

2.1.2. *Bad* Thematic Condition

Retrieving data:

```
nl_en_bad <- read.csv('LSTM_1_13363796_nl_en_bad.csv') %>%
  select(prime_type, target_type, prime_nr, target_nr, surp)
```

```
##   prime_type target_type prime_nr target_nr    surp
## 1   nl_unamb    nl_unamb        0         0  8.4569
## 2   nl_unamb    nl_unamb        0         1  9.0207
## 3   nl_unamb    nl_unamb        0         2 14.3938
## 4   nl_unamb    nl_unamb        0         3 12.4891
## 5   nl_unamb    nl_unamb        0         4 14.4023
## 6   nl_unamb    nl_unamb        0         5 10.3669
```

Calculating garden-path effect with *calculate_surpdiffs_2* function for each priming condition:

```
# Prime type: Ambiguous

nl_en_bad_1 <- calculate_surpdiffs_2(nl_en_bad, 'nl_amb', 'nl_amb', 'nl_unamb', 'nl')
nl_en_bad_2 <- calculate_surpdiffs_2(nl_en_bad, 'nl_amb', 'en_amb', 'en_unamb', 'nl')
nl_en_bad_3 <- calculate_surpdiffs_2(nl_en_bad, 'en_amb', 'nl_amb', 'nl_unamb', 'nl')
nl_en_bad_4 <- calculate_surpdiffs_2(nl_en_bad, 'en_amb', 'en_amb', 'en_unamb', 'nl')

# Prime type: Unambiguous

nl_en_bad_5 <- calculate_surpdiffs_2(nl_en_bad, 'nl_unamb', 'nl_amb', 'nl_unamb', 'nl')
nl_en_bad_6 <- calculate_surpdiffs_2(nl_en_bad, 'nl_unamb', 'en_amb', 'en_unamb', 'nl')
nl_en_bad_7 <- calculate_surpdiffs_2(nl_en_bad, 'en_unamb', 'nl_amb', 'nl_unamb', 'nl')
nl_en_bad_8 <- calculate_surpdiffs_2(nl_en_bad, 'en_unamb', 'en_amb', 'en_unamb', 'nl')
```

Combining the results from all the priming conditions:

```
nl_en_bad_results <- rbind(nl_en_bad_1, nl_en_bad_2, nl_en_bad_3, nl_en_bad_4,
                           nl_en_bad_5, nl_en_bad_6, nl_en_bad_7, nl_en_bad_8)
```

```
##   surpdiffs prime_type language_pair target_prof
## 1    0.6131  Ambiguous        Within          L1
## 2    0.0940  Ambiguous        Within          L1
## 3    0.4965  Ambiguous        Within          L1
## 4    0.3295  Ambiguous        Within          L1
## 5    0.7270  Ambiguous        Within          L1
## 6    0.6185  Ambiguous        Within          L1
```

2.2. L1 English - L2 Dutch Model

2.2.1. *Good* Thematic Condition

Retrieving data:

```
en_nl_good <- read.csv('LSTM_1_13262074_en_nl_good.csv') %>%
  select(prime_type, target_type, prime_nr, target_nr, surp)
```

```
##   prime_type target_type prime_nr target_nr    surp
## 1   nl_unamb    nl_unamb        0         0  9.6445
## 2   nl_unamb    nl_unamb        0         1  9.9807
## 3   nl_unamb    nl_unamb        0         2  9.7834
## 4   nl_unamb    nl_unamb        0         3 11.8249
## 5   nl_unamb    nl_unamb        0         4 11.9737
## 6   nl_unamb    nl_unamb        0         5 12.0899
```

Calculating garden-path effect with *calculate_surpdiffs_2* function for each priming condition:

```
# Prime type: Ambiguous

en_nl_good_1 <- calculate_surpdiffs_2(en_nl_good, 'nl_amb', 'nl_amb', 'nl_unamb', 'en')
en_nl_good_2 <- calculate_surpdiffs_2(en_nl_good, 'nl_amb', 'en_amb', 'en_unamb', 'en')
en_nl_good_3 <- calculate_surpdiffs_2(en_nl_good, 'en_amb', 'nl_amb', 'nl_unamb', 'en')
en_nl_good_4 <- calculate_surpdiffs_2(en_nl_good, 'en_amb', 'en_amb', 'en_unamb', 'en')

# Prime type: Unambiguous

en_nl_good_5 <- calculate_surpdiffs_2(en_nl_good, 'nl_unamb', 'nl_amb', 'nl_unamb', 'en'
en_nl_good_6 <- calculate_surpdiffs_2(en_nl_good, 'nl_unamb', 'en_amb', 'en_unamb', 'en'
en_nl_good_7 <- calculate_surpdiffs_2(en_nl_good, 'en_unamb', 'nl_amb', 'nl_unamb', 'en'
en_nl_good_8 <- calculate_surpdiffs_2(en_nl_good, 'en_unamb', 'en_amb', 'en_unamb', 'en'
```

Combining the results from all the priming conditions:

```
en_nl_good_results <- rbind(en_nl_good_1, en_nl_good_2, en_nl_good_3, en_nl_good_4,
                            en_nl_good_5, en_nl_good_6, en_nl_good_7, en_nl_good_8)
```

```
##   surpdiffs prime_type language_pair target_prof
## 1    0.3024  Ambiguous        Within          L2
## 2    0.1464  Ambiguous        Within          L2
## 3   -0.1492  Ambiguous        Within          L2
## 4   -0.0375  Ambiguous        Within          L2
## 5   -0.2162  Ambiguous        Within          L2
## 6    0.7067  Ambiguous        Within          L2
```

2.2.2. *Bad* Thematic Condition

Retrieving data:

```
en_nl_bad <- read.csv('LSTM_1_13262074_en_nl_bad.csv') %>%
  select(prime_type, target_type, prime_nr, target_nr, surp)
```

```
##   prime_type target_type prime_nr target_nr    surp
## 1   nl_unamb    nl_unamb        0         0  9.0927
## 2   nl_unamb    nl_unamb        0         1  9.2318
## 3   nl_unamb    nl_unamb        0         2 12.9569
## 4   nl_unamb    nl_unamb        0         3 11.5254
## 5   nl_unamb    nl_unamb        0         4 12.7232
## 6   nl_unamb    nl_unamb        0         5 10.5949
```

Calculating garden-path effect with *calculate_surpdiffs_2* function for each priming condition:

```
# Prime type: Ambiguous

en_nl_bad_1 <- calculate_surpdiffs_2(en_nl_bad, 'nl_amb', 'nl_amb', 'nl_unamb', 'en')
en_nl_bad_2 <- calculate_surpdiffs_2(en_nl_bad, 'nl_amb', 'en_amb', 'en_unamb', 'en')
en_nl_bad_3 <- calculate_surpdiffs_2(en_nl_bad, 'en_amb', 'nl_amb', 'nl_unamb', 'en')
en_nl_bad_4 <- calculate_surpdiffs_2(en_nl_bad, 'en_amb', 'en_amb', 'en_unamb', 'en')

# Prime type: Unambiguous

en_nl_bad_5 <- calculate_surpdiffs_2(en_nl_bad, 'nl_unamb', 'nl_amb', 'nl_unamb', 'en')
en_nl_bad_6 <- calculate_surpdiffs_2(en_nl_bad, 'nl_unamb', 'en_amb', 'en_unamb', 'en')
en_nl_bad_7 <- calculate_surpdiffs_2(en_nl_bad, 'en_unamb', 'nl_amb', 'nl_unamb', 'en')
en_nl_bad_8 <- calculate_surpdiffs_2(en_nl_bad, 'en_unamb', 'en_amb', 'en_unamb', 'en')
```

Combining the results from all the priming conditions:

```
en_nl_bad_results <- rbind(nl_en_bad_1, nl_en_bad_2, nl_en_bad_3, nl_en_bad_4, nl_en_bad
```

```
##   surpdiffs prime_type language_pair target_prof
## 1    0.6131  Ambiguous        Within          L1
## 2    0.0940  Ambiguous        Within          L1
## 3    0.4965  Ambiguous        Within          L1
## 4    0.3295  Ambiguous        Within          L1
## 5    0.7270  Ambiguous        Within          L1
## 6    0.6185  Ambiguous        Within          L1
```

2.3. Combining results from L1 Dutch - L2 English Model and L1 English - L2 Dutch Model

2.3.1. *Good* Thematic Condition

```
exper2_good_results <- rbind(nl_en_good_results, en_nl_good_results)
```

```
##   surpdiffs prime_type language_pair target_prof
## 1    0.0060  Ambiguous        Within          L1
## 2   -0.2901  Ambiguous        Within          L1
## 3   -0.2153  Ambiguous        Within          L1
## 4    0.0465  Ambiguous        Within          L1
## 5   -0.2341  Ambiguous        Within          L1
## 6    0.4800  Ambiguous        Within          L1
```

2.3.1. *Bad* Thematic Condition

```
exper2_bad_results <- rbind(nl_en_bad_results, en_nl_bad_results)
```

```
##   surpdiffs prime_type language_pair target_prof
## 1    0.6131  Ambiguous        Within          L1
## 2    0.0940  Ambiguous        Within          L1
## 3    0.4965  Ambiguous        Within          L1
## 4    0.3295  Ambiguous        Within          L1
## 5    0.7270  Ambiguous        Within          L1
## 6    0.6185  Ambiguous        Within          L1
```

III Plots

In order to plot the results, we first need to summarize them. Below is the format of converting the grouping variables *IV_1*, *IV_2* of a dataframe *df* to factors, and calculating the means and confidence intervals of the dependent variable *DV* (i.e., surprisal differences) on the independent variables.

```
df %>%
  mutate(IV_1 = factor(IV_1),
         IV_2 = factor(IV_2)) %>%
  group_by(IV_1, IV_2) %>%
  summarise(
    DV_mean = mean(DV),
    DV_ci   = 1.96 * sd(DV)/sqrt(n())) # standard error of the mean, and multiplying th
                                       #the distance from the mean to the 95% interval
                                       #(which we then add or subtract from the mean la

    )
  }
```

Finally, below is the format of plotting the summarized data in which *df_summary* is a dataframe of the summarized data, *x_label* and *y_label* indicate what x-axis and y-axis denote which correspond to one of the independent variables, *IV_1* or *IV_2*, and the dependent variable, *DV*. *DV_mean* and *DV_ci* refer to the column names that have means and confidence intervals of the dependent variable. Lastly, the title of the graph consists of *title_line_1* and *title_line_2* which refer to the first line and second line of the title, respectively. *legend_title* is the title of the legend which specifies the grouping variable.

```
df_summary %>%
  ggplot(aes(x = IV_1, y = DV_mean, group = IV_2)) +
  geom_line(aes(linetype = IV_2)) +
  geom_errorbar(aes(ymin = DV_mean - DV_ci, ymax = DV_mean + DV_ci),
                width = .1, linetype = 1) +
  geom_point(size = 2) +
  geom_point(size = 1, color = "white") +
  guides(linetype = guide_legend(legend)) +
  labs(title = paste(title_line_1,
                     title_line_2,
                     " ",
                     sep = "\n"),
       caption = paste(" ",
                       "Error bar: 95% CI",
                       sep = "\n"),
       x = x_label,
       y = y_label) +
  theme(
    legend.title = element_text(family = size = 10),
    panel.background = element_rect(fill = "white"),
    legend.key  = element_rect(fill = "white"),
    axis.line.x = element_line(colour = "black", size = 1),
    axis.line.y = element_line(colour = "black", size = 1))
```

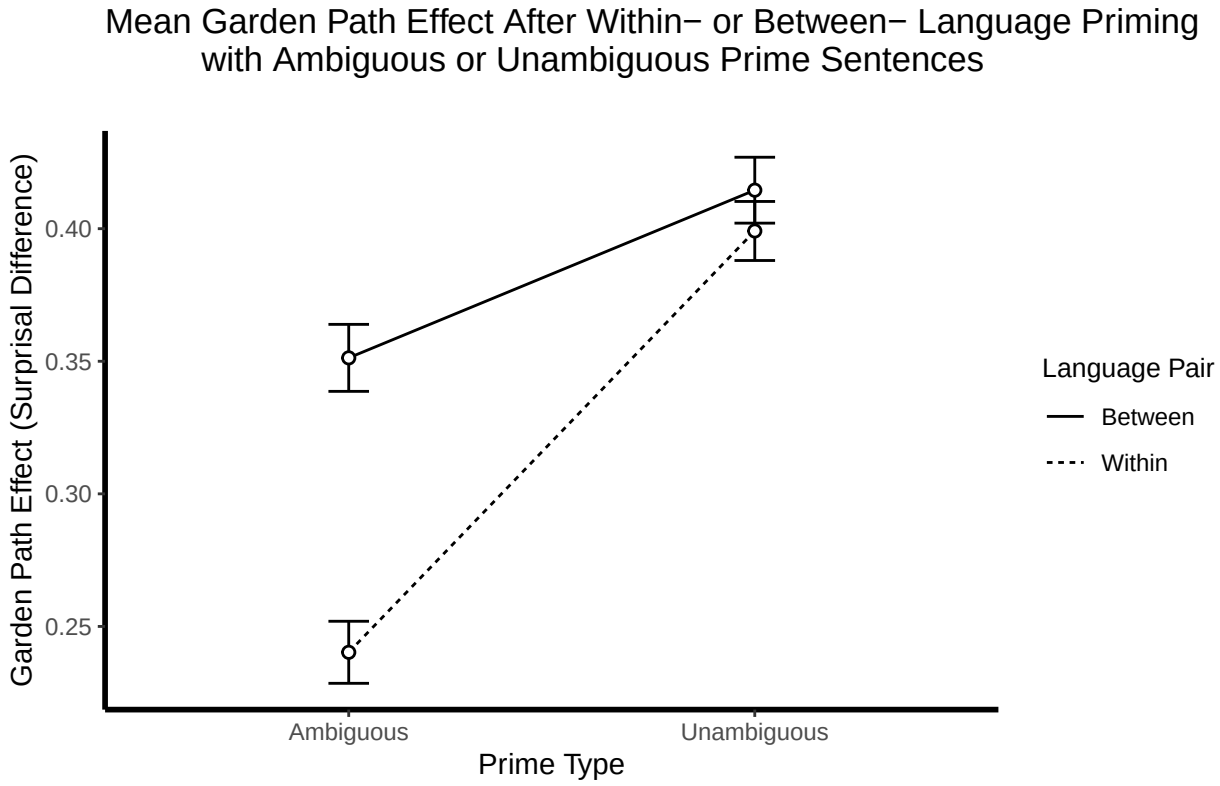
1. First Experiment

1.1. *Good* Thematic Condition

Data summary:

```
exper1_good_summary <- exper1_good_results %>%
  mutate(prime_type = factor(prime_type),
         language_pair = factor(language_pair)) %>%
  group_by(prime_type, language_pair) %>%
  summarise(
    surpdiffs_mean = mean(surpdiffs),
    surpdiffs_ci   = 1.96 * sd(surpdiffs)/sqrt(n()))
```

```
## # A tibble: 4 x 4
## # Groups:   prime_type [2]
##   prime_type  language_pair surpdiffs_mean surpdiffs_ci
##   <fct>       <fct>                  <dbl>        <dbl>
## 1 Ambiguous   Between                0.351       0.0126
## 2 Ambiguous   Within                 0.240       0.0117
## 3 Unambiguous Between                0.415       0.0124
## 4 Unambiguous Within                 0.399       0.0111
```



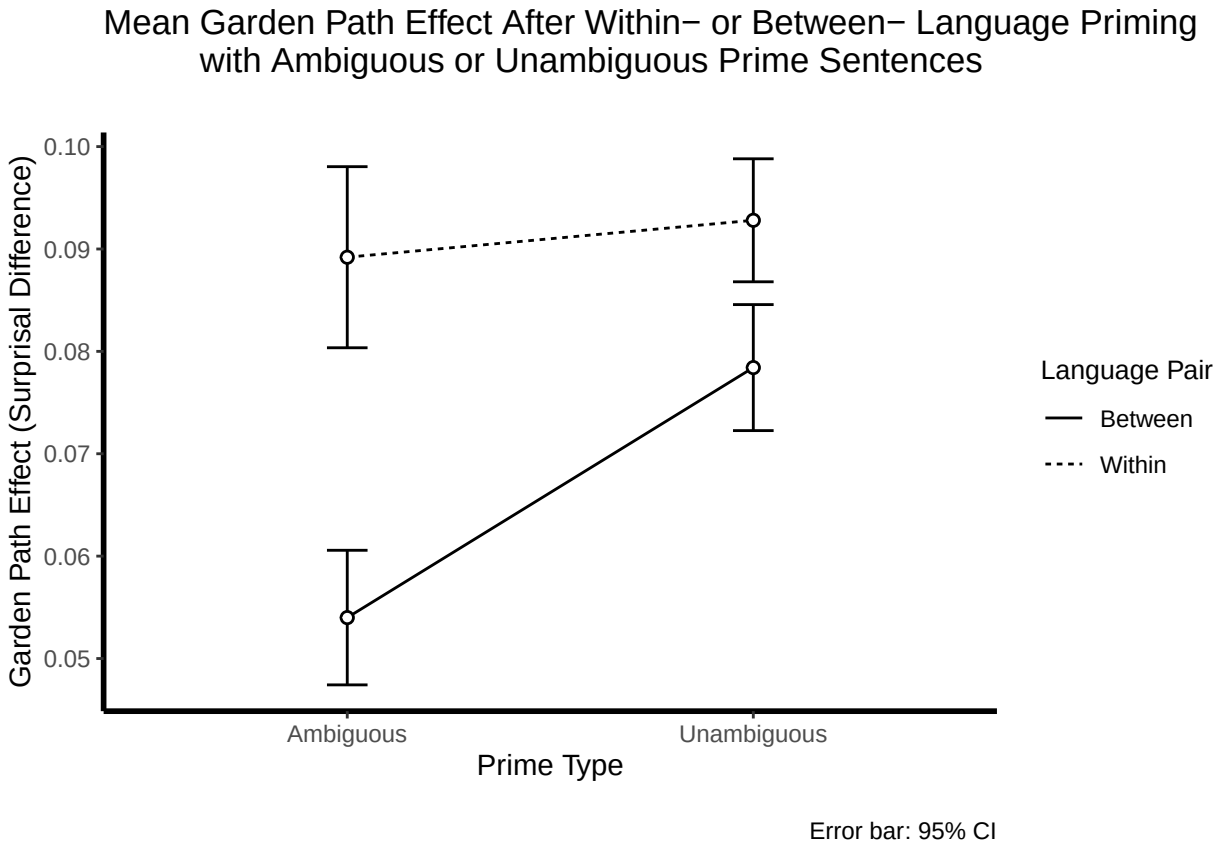
Plot:

1.2. *Bad* Thematic Condition

```
exper1_bad_summary <- exper1_bad_results %>%
  mutate(prime_type = factor(prime_type),
         language_pair = factor(language_pair)) %>%
  group_by(prime_type, language_pair) %>%
  summarise(
```

```
    surpdiffs_mean = mean(surpdiffs),
    surpdiffs_ci   = 1.96 * sd(surpdiffs)/sqrt(n()))
```

```
## # A tibble: 4 x 4
## # Groups:   prime_type [2]
##   prime_type  language_pair surpdiffs_mean surpdiffs_ci
##   <fct>       <fct>                  <dbl>        <dbl>
## 1 Ambiguous   Between               0.0540      0.00657
## 2 Ambiguous   Within                0.0892      0.00884
## 3 Unambiguous Between               0.0784      0.00615
## 4 Unambiguous Within                0.0928      0.00601
```



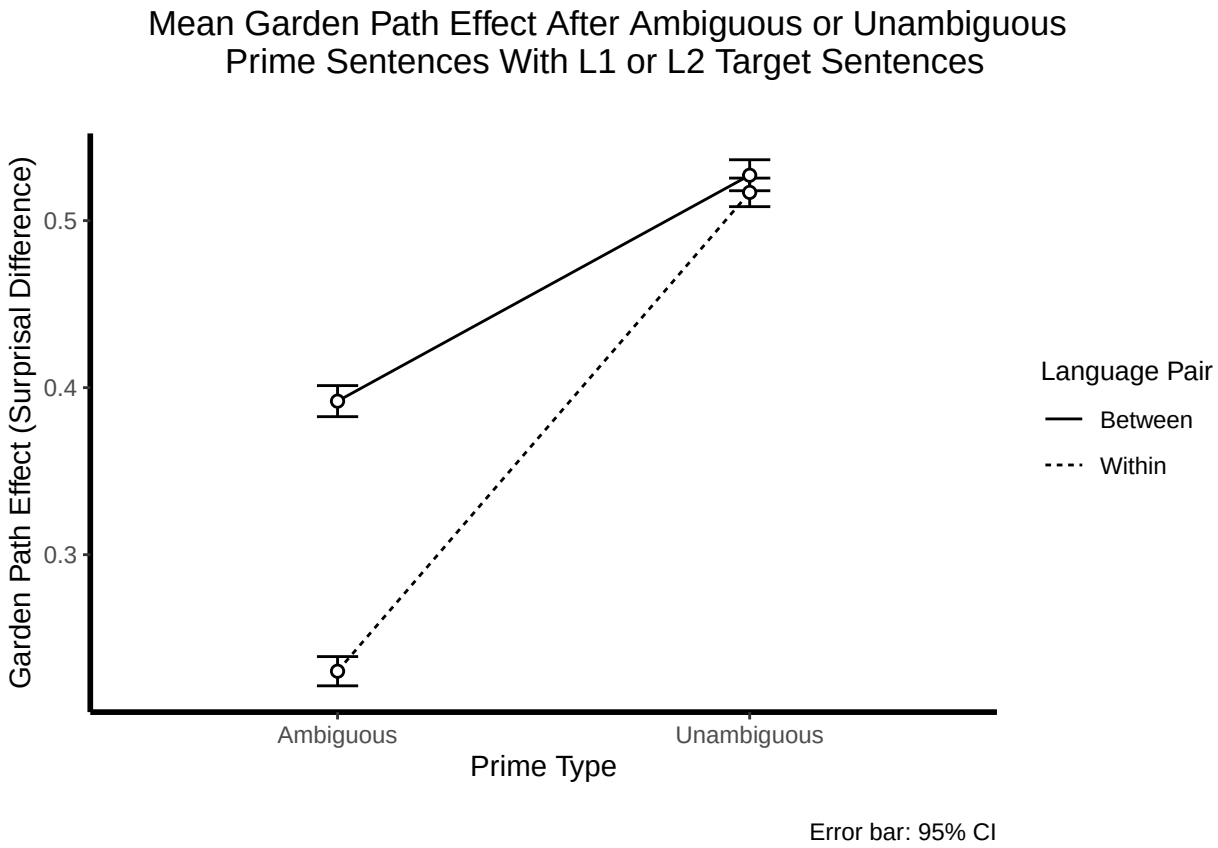
2. Second Experiment

2.1. The Effect of Language Pair

2.1.1. *Good* Thematic Condition

```
exper2_good_summary_1 <- exper2_good_results %>%
  mutate(prime_type = factor(prime_type),
         language_pair = factor(language_pair)) %>%
  group_by(prime_type, language_pair) %>%
  summarise(
    surpdiffs_mean = mean(surpdiffs),
    surpdiffs_ci   = 1.96 * sd(surpdiffs)/sqrt(n()))
```

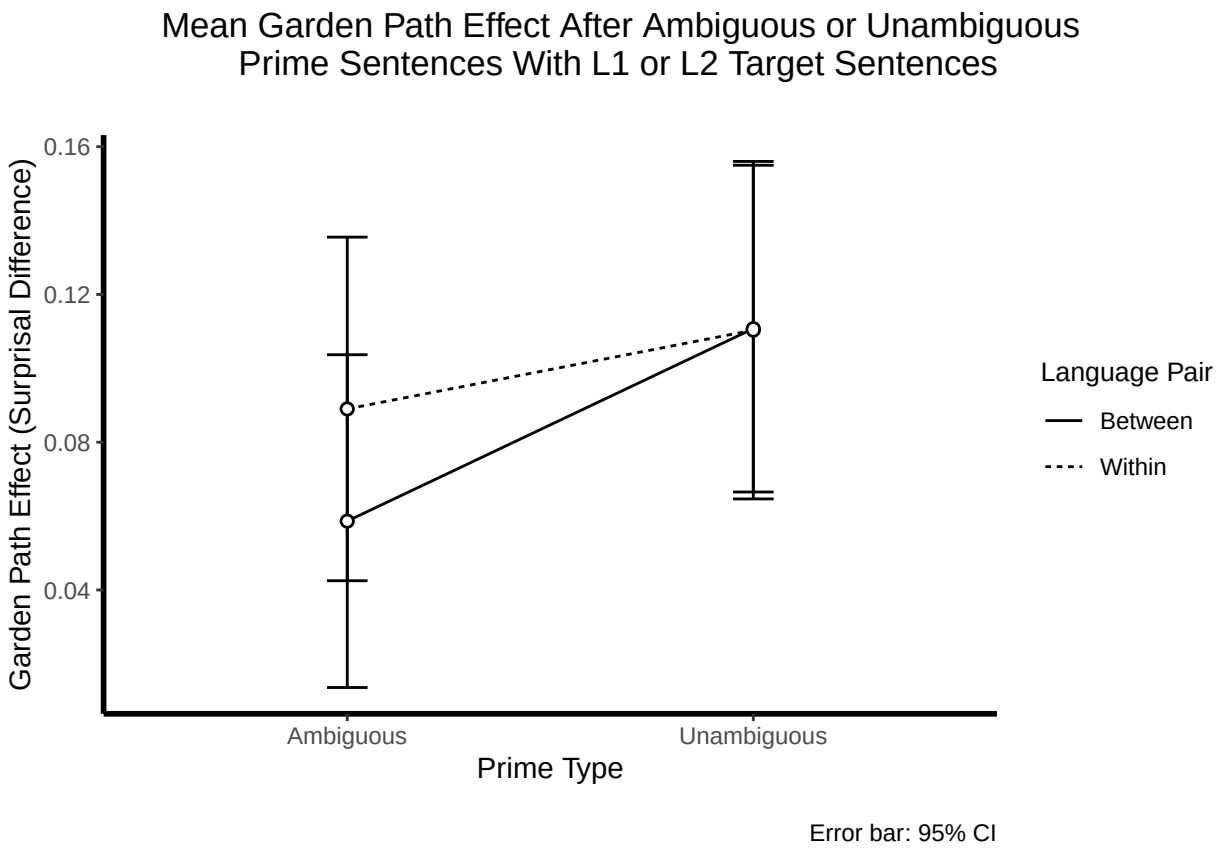
```
## # A tibble: 4 x 4
## # Groups:   prime_type [2]
##   prime_type  language_pair surpdiffs_mean surpdiffs_ci
##   <fct>       <fct>                  <dbl>        <dbl>
## 1 Ambiguous   Between                0.392      0.00930
## 2 Ambiguous   Within                 0.230      0.00875
## 3 Unambiguous Between                0.527      0.00928
## 4 Unambiguous Within                 0.517      0.00854
```



2.1.2. *Bad* Thematic Condition

```
exper2_bad_summary_1 <- exper2_bad_results %>%
  mutate(prime_type = factor(prime_type),
         language_pair = factor(language_pair)) %>%
  group_by(prime_type, language_pair) %>%
  summarise(
    surpdiffs_mean = mean(surpdiffs),
    surpdiffs_ci   = 1.96 * sd(surpdiffs)/sqrt(n()))
```

```
## # A tibble: 4 x 4
## # Groups:   prime_type [2]
##   prime_type  language_pair surpdiffs_mean surpdiffs_ci
##   <fct>       <fct>                 <dbl>        <dbl>
## 1 Ambiguous   Between              0.0586       0.0451
## 2 Ambiguous   Within               0.0890       0.0465
## 3 Unambiguous Between              0.111        0.0442
## 4 Unambiguous Within               0.110        0.0457
```



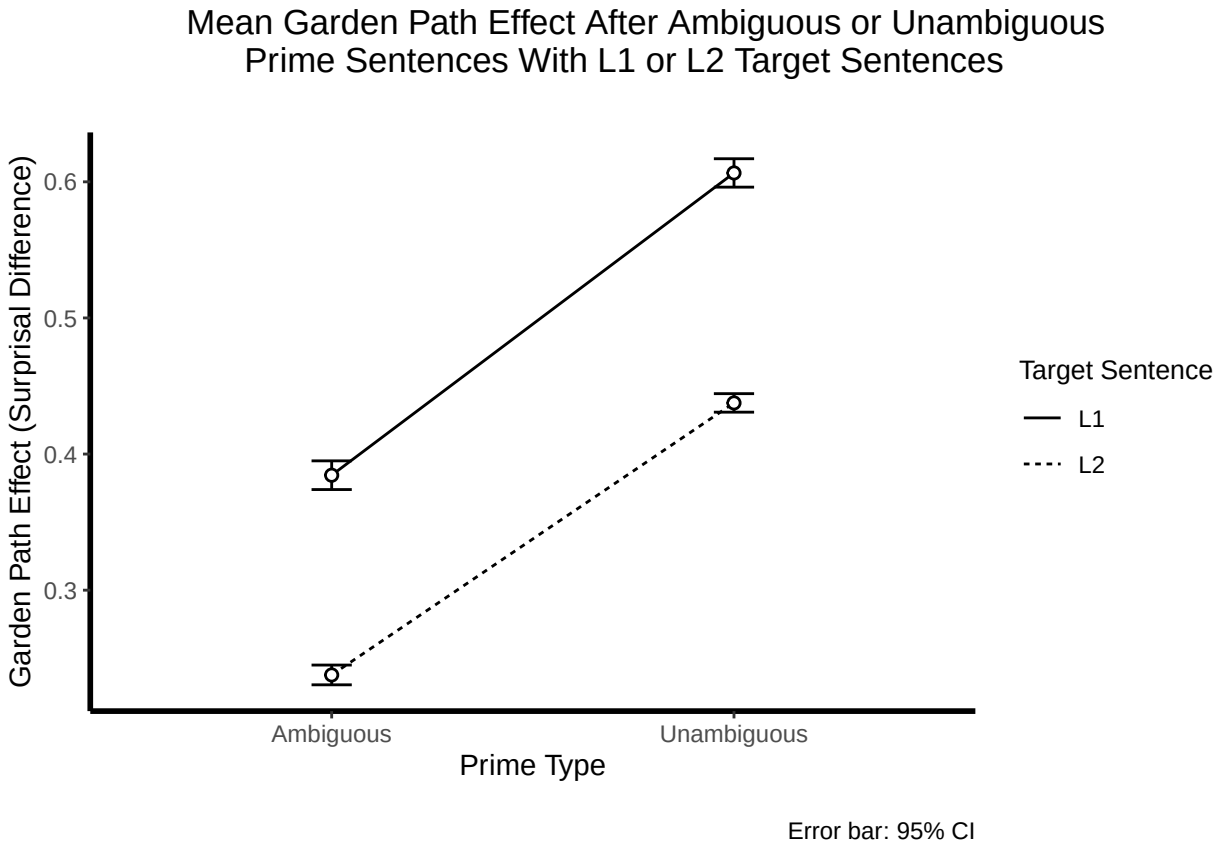
2.2. The Effect of Target Sentence Proficiency

2.2.1. *Good* Thematic Condition

```
exper2_good_summary_2 <- exper2_good_results %>%
  mutate(prime_type = factor(prime_type),
         target_prof = factor(target_prof)) %>%
  group_by(prime_type, target_prof) %>%
  summarise(
    surpdiffs_mean = mean(surpdiffs),
    surpdiffs_ci   = 1.96 * sd(surpdiffs)/sqrt(n()))
```

```
## # A tibble: 4 x 4
## # Groups:   prime_type [2]
```

```
##   prime_type  target_prof surpdiffs_mean surpdiffs_ci
##   <fct>       <fct>                <dbl>        <dbl>
## 1 Ambiguous   L1                   0.384      0.0105
## 2 Ambiguous   L2                   0.238      0.00726
## 3 Unambiguous L1                   0.607      0.0104
## 4 Unambiguous L2                   0.438      0.00680
```



2.2.2. *Bad* Thematic Condition

```
exper2_bad_summary_2 <- exper2_bad_results %>%
  mutate(prime_type = factor(prime_type),
         target_prof = factor(target_prof)) %>%
  group_by(prime_type, target_prof) %>%
  summarise(
    surpdiffs_mean = mean(surpdiffs),
    surpdiffs_ci   = 1.96 * sd(surpdiffs)/sqrt(n()))
```

```
## # A tibble: 4 x 4
## # Groups:   prime_type [2]
##   prime_type  target_prof surpdiffs_mean surpdiffs_ci
##   <fct>       <fct>                <dbl>        <dbl>
## 1 Ambiguous   L1                  0.120       0.00636
## 2 Ambiguous   L2                  0.0274      0.0644
## 3 Unambiguous L1                  0.208       0.00520
## 4 Unambiguous L2                  0.0129      0.0633
```