sound_symbolism

December 22, 2023

1 SISSA - Language Reading and The Brain - a.y. 2023-24

2 Sebastiano Quintavalle

2.1 1. Introduction

The purpose of this notebook is to analyze the results of an experiment carried out during an academic class, collected in the file `sound_symbolism.csv` (in order to automatically download the file from url, it is required to have first accessed basecamp in the browser and possibly updated the url in the `settings.py` file).

The **22 students** in the class were made to listen to **32 words** in a language unfamiliar to them. These words referred to objects of **small size** or **large size**, and for each example the student was asked to indicate one of the two options.

2.1.1 Data Preparation

Download the file and store it to a local directory.

[2]:
```
from settings import DATA_URL, DATA_DIR, FILE_NAME
from io_ import DataDownloader

downloader = DataDownloader(url=DATA_URL, dir_path=DATA_DIR, file_name=FILE_NAME)

downloader
```

[2]:
```
DataDownloader[file: C:\Users\user.LAPTOP-
G27BJ7JO\Desktop\SISSA\CREPALDI\assigment1\data\sound_symbolism.csv; downloaded:
True]
```

[3]:
```
downloader.download()
```

```
Directory already exists:
C:\Users\user.LAPTOP-G27BJ7JO\Desktop\SISSA\CREPALDI\assigment1\data
File downloaded at: C:\Users\user.LAPTOP-
G27BJ7JO\Desktop\SISSA\CREPALDI\assigment1\data\sound_symbolism.csv
```

Once the file is downloaded, we load it as a Data Frame.

[4]:
```
from io_ import CSVLoader

loader = CSVLoader(file_path=downloader.file_path)
loader
```

[4]:
```
CSVLoader[file: C:\Users\user.LAPTOP-
G27BJ7JO\Desktop\SISSA\CREPALDI\assigment1\data\sound_symbolism.csv]
```

[5]:
```
data = loader.load()
```

```
Loading C:\Users\user.LAPTOP-
G27BJ7JO\Desktop\SISSA\CREPALDI\assigment1\data\sound_symbolism.csv ...
Complete
```

Let's inspect the first few rows.

[6]:
```
data.head(10)
```

[6]:

	word_id	sbj_id	response	word	language	meaning	is_sound_symbolic
0	word1	sbj1	big	dev	turkish	big	no
1	word1	sbj2	small	dev	turkish	big	no
2	word1	sbj3	big	dev	turkish	big	no
3	word1	sbj4	small	dev	turkish	big	no
4	word1	sbj5	small	dev	turkish	big	no
5	word1	sbj6	small	dev	turkish	big	no
6	word1	sbj7	small	dev	turkish	big	no
7	word1	sbj8	small	dev	turkish	big	no
8	word1	sbj9	big	dev	turkish	big	no
9	word1	sbj10	big	dev	turkish	big	no

2.2 2. Analysis

2.2.1 Disclaimer

With the aim of making the analysis more linear in the notebook, the data were converted from a table form and a class architecture implemented and documented in `model.py`. This allows for clearer data modeling and allows one to focus solely on the results by demanding the implementation details to the classes.

2.2.2 Experiment words

First we create the collection of words used during the experiment.

[7]:
```
from model import Meaning, Word, Words

words_experiment = Words()

for row in data.loc[:, ['word_id', 'word', 'language', 'meaning',␣
 ↪'is_sound_symbolic']].drop_duplicates().itertuples(index=False):
```

```
    meaning = Meaning.from_string(row.meaning)
    is_sound_symbolic = row.is_sound_symbolic == "yes"

    new_word = Word(
        id_=row.word_id,
        word=row.word,
        language=row.language,
        meaning=meaning,
        is_sound_symbolic=is_sound_symbolic
    )

    words_experiment.add_word(word=new_word)
```

Words are equally distributed by large and small.

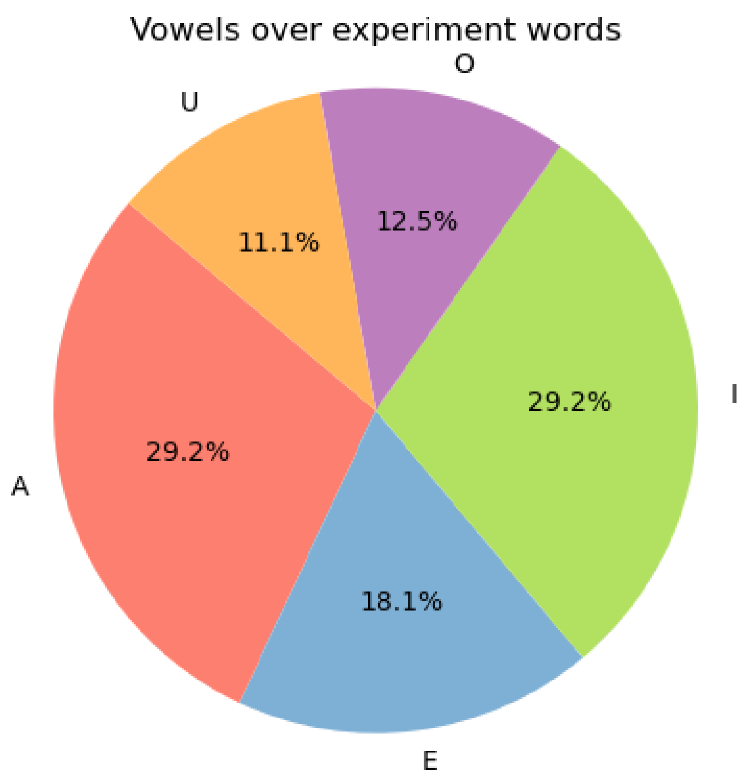
[8]:
```
words_experiment
```

[8]:
```
Words[count: 32, small: 16, big: 16]
```

We can identify two simple characteristics for a set of words, **average length** and **vowel distribution**.

The distribution of vowels is fairly balanced in the word set.

[9]:
```
words_experiment.plot_vowels_distr(title="Vowels over experiment words")
```



The average length.

[10]:
```
print(f"Average length: {words_experiment.avg_length}")
```

```
Average length: 5.375
```

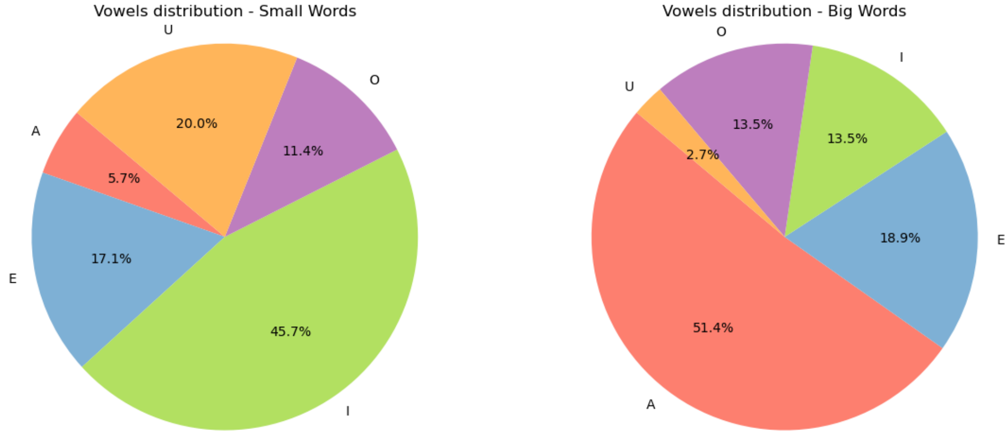
Let's separate the words into small and big.

[11]:
```
words_experiment_small, words_experiment_big = words_experiment.
↪get_meaning_split()

print(words_experiment_small)
print(words_experiment_big)
```

```
Words[count: 16, small: 16, big: 0]
Words[count: 16, small: 0, big: 16]
```

The distribution of vowels between the two types of meanings shows a clear pattern: words referring to **something small** tend to make predominant use of the vowel ***I***, while words referring to **something large** make predominant use of ***A***. Intuitively, this may be motivated by the fact that the *A* is an **open vowel**, that is, it is produced with as much space as possible between the tongue and palate and may therefore refer to a **concept of extension**. In contrast, the **closed sound** of the *I* may somehow evoke something **smaller in size**.

[12]:
```
from matplotlib import pyplot as plt

fig, axes = plt.subplots(1, 2, figsize=(12, 5))
words_experiment_small.plot_vowels_distr(ax=axes[0], title="Vowels distribution␣
↪- Small Words")
words_experiment_big.  plot_vowels_distr(ax=axes[1], title="Vowels distribution␣
↪- Big Words"  )

plt.tight_layout()
plt.show()
```



The average length does not reveal a clear pattern.

[13]:
```
print("Average length:")
print(f" - Small words: {words_experiment_small.avg_length}")
print(f" - Big   words: {words_experiment_big  .avg_length}")
```

```
Average length:
 - Small words: 5.125
 - Big   words: 5.625
```

2.2.3 Subjects

Let's add **experiment information**. The set of responses given by a single subject is considered as an experiment. The experiments are then grouped into a single collection.

[14]:
```
from model import Subject, Experiments

experiments = Experiments(words=words_experiment)

subject_ids = data.loc[:, "sbj_id"].unique().tolist()

for subject_id in subject_ids:

    # Create new subject
    new_subject = Subject(id_=subject_id)

    # Add answers
    for row in data[data.loc[:, "sbj_id"] == subject_id].itertuples(index=False):

        new_subject.add_answer(
            word_id=row.word_id,
            answer=Meaning.from_string(s=row.response)
```

```
        )

    experiments.add_subject(subject=new_subject)
```

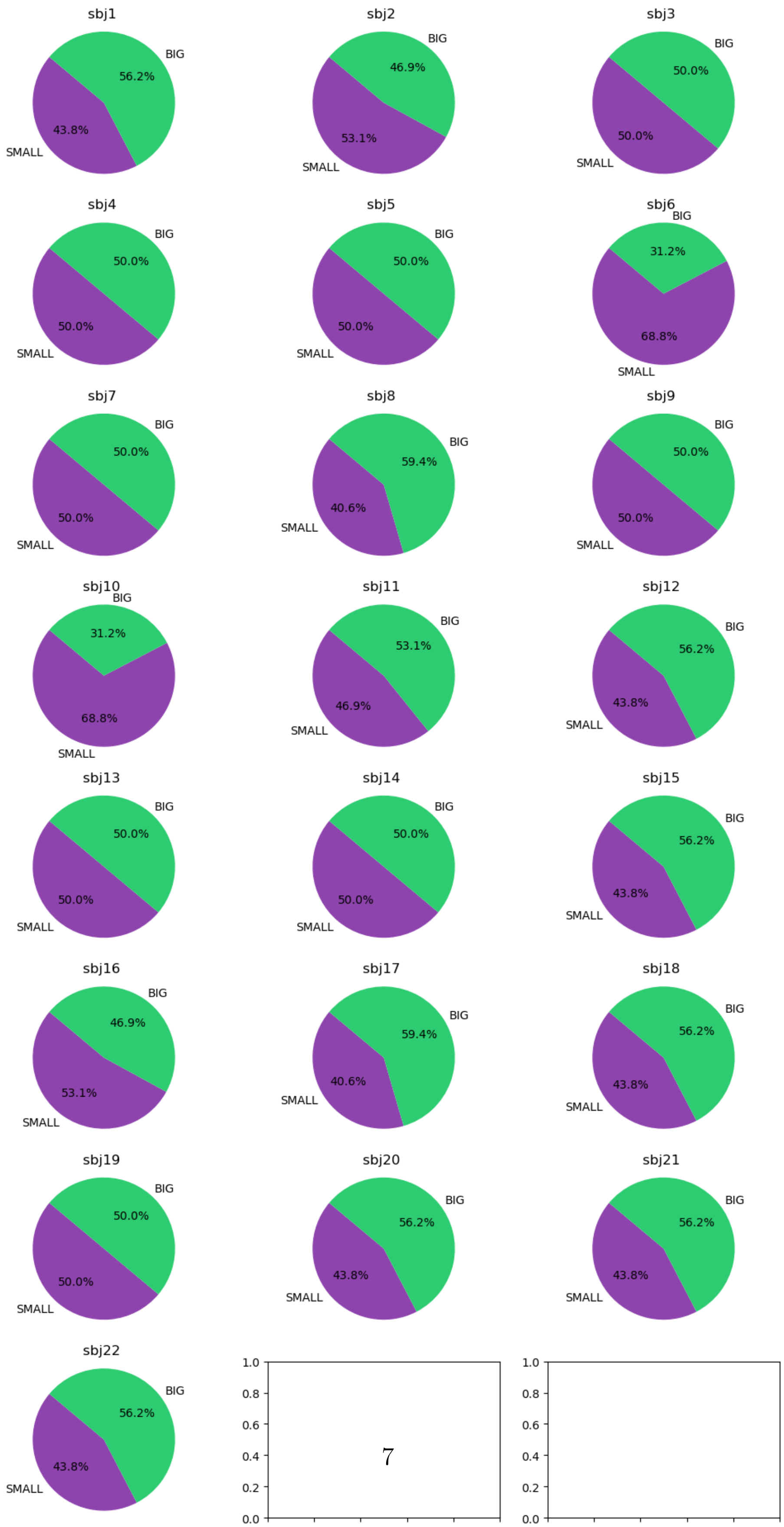
[15]:
```
experiments
```

[15]: Experiment[Subjects: 22 - Mean score: 0.646]

We first analyze the proportion of small vs large responses given by each subject. The distribution of responses is **fairly balanced for each subject**. Thus we can think they had a notion of the distribution of words in the experimental set, that was also balanced.

[16]:
```
import matplotlib.pyplot as plt

N_COL = 3
N_ROW = 8

fig, axes = plt.subplots(N_ROW, N_COL, figsize=(10, 21))

for i in range(N_COL * N_ROW):

    if i >= len(experiments):
        break

    row = i // N_COL
    col = i % N_COL

    experiments[subject_ids[i]].subject.plot_answer_count(ax=axes[row, col],␣
↪title="")

# Add a main title
fig.suptitle("Subject answer proportions", fontsize=24)

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```

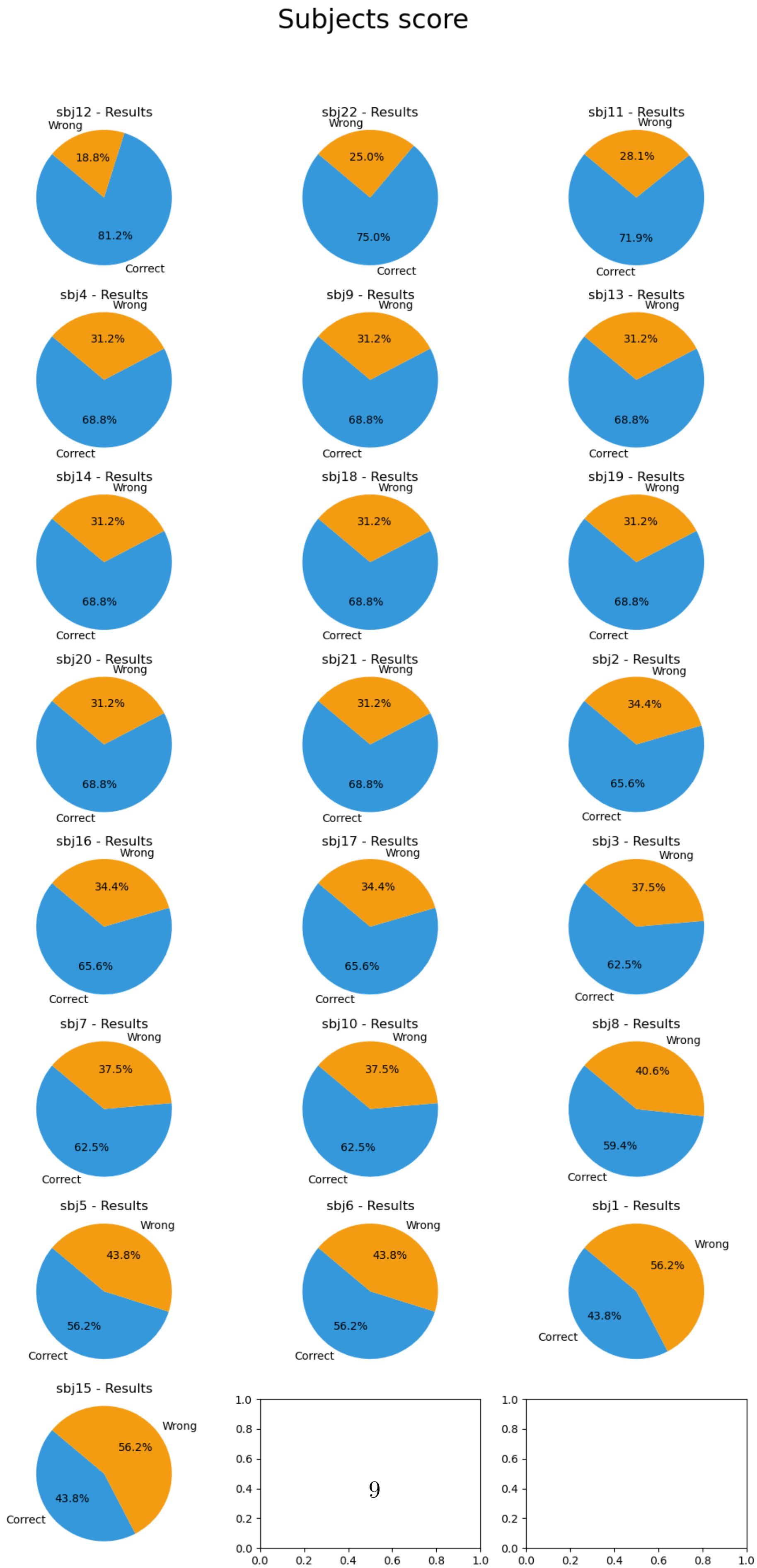
Subject answer proportions



2.2.4 Subject Scores

Let's now consider the subjects' responses based on the true meaning of the word and calculate the **percentage of correct answers**.

[17]:
```
experiments.plot_subjects_scores(n_row=8, by_score=True)
```


Subjects score
sbj12 - Results
Wrong
18.8%
81.2%
Correct
sbj22 - Results
Wrong
25.0%
75.0%
Correct
sbj11 - Results
Wrong
28.1%
71.9%
Correct
sbj4 - Results
Wrong
31.2%
68.8%
Correct
sbj9 - Results
Wrong
31.2%
68.8%
Correct
sbj13 - Results
Wrong
31.2%
68.8%
Correct
sbj14 - Results
Wrong
31.2%
68.8%
Correct
sbj18 - Results
Wrong
31.2%
68.8%
Correct
sbj19 - Results
Wrong
31.2%
68.8%
Correct
sbj20 - Results
Wrong
31.2%
68.8%
Correct
sbj21 - Results
Wrong
31.2%
68.8%
Correct
sbj2 - Results
Wrong
34.4%
65.6%
Correct
sbj16 - Results
Wrong
34.4%
65.6%
Correct
sbj17 - Results
Wrong
34.4%
65.6%
Correct
sbj3 - Results
Wrong
37.5%
62.5%
Correct
sbj7 - Results
Wrong
37.5%
62.5%
Correct
sbj10 - Results
Wrong
37.5%
62.5%
Correct
sbj8 - Results
Wrong
40.6%
59.4%
Correct
sbj5 - Results
Wrong
43.8%
56.2%
Correct
sbj6 - Results
Wrong
43.8%
56.2%
Correct
sbj1 - Results
Wrong
56.2%
43.8%
Correct
sbj15 - Results
Wrong
56.2%
43.8%
Correct
1.0
0.8
0.6
0.4
0.2
0.0
0.0 0.2 0.4 0.6 0.8 1.0
1.0
0.8
0.6
0.4
0.2
0.0
0.0 0.2 0.4 0.6 0.8 1.0

The balanced distribution of meanings in words allows us to establish a baseline at 50%, achieved by a subject who doesn't listen to the words but simply guesses for each example. On average, **subjects perform 10% above the baseline**.

[18]:
```
print(f"Mean score: {round(experiments.mean_score, 3)}")
```

```
Mean score: 0.646
```

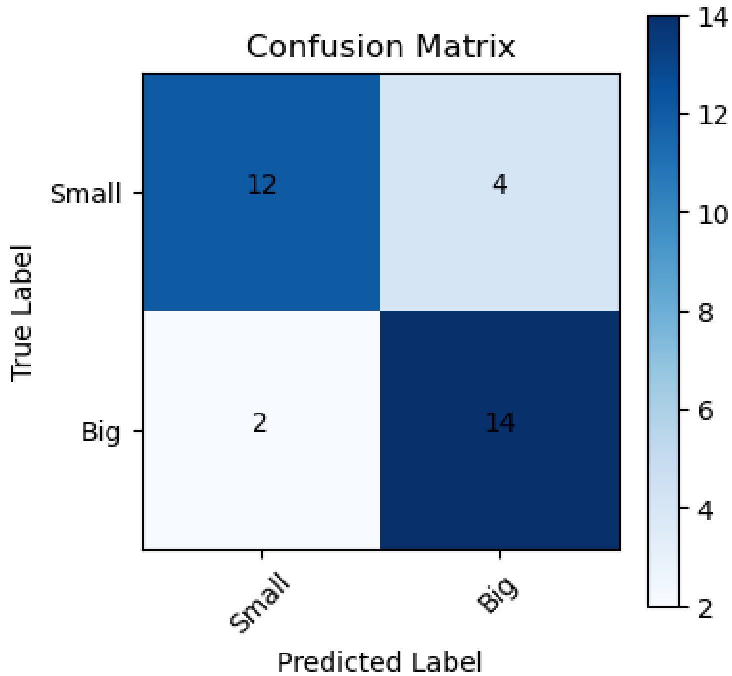
The subject with the **highest score** achieved an **excellent result** of 80%, well above the average, while the one with the **lowest score** is only **slightly below the baseline**.

[19]:
```
_, best  = experiments.sort_by_score()[0]
_, worst = experiments.sort_by_score()[-1]

print(f"Best  subject: {best} ")
print(f"Worst subject: {worst}")
```

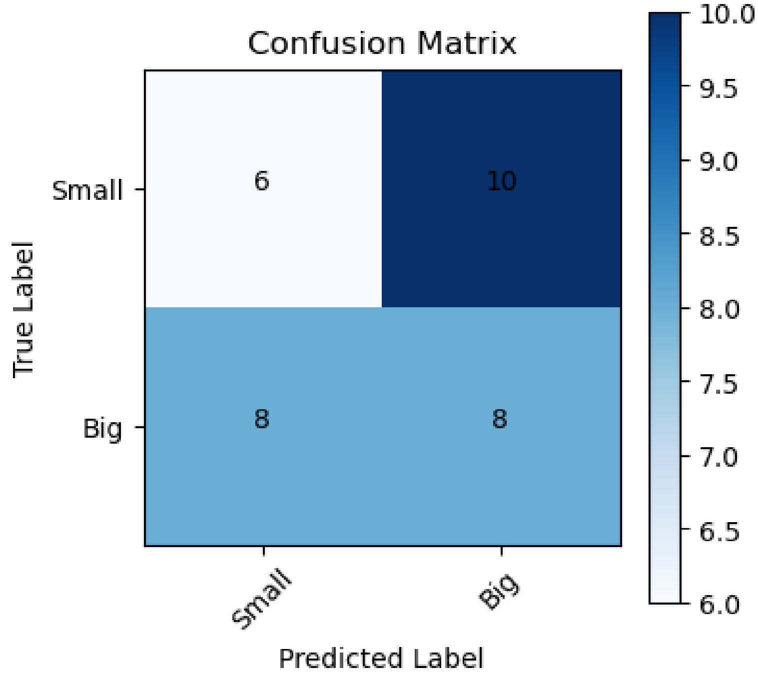
```
Best  subject: Experiment sbj12[Score: 0.8125]
Worst subject: Experiment sbj15[Score: 0.4375]
```

Let's inspect the confusion matrixes of the two subjects.

[20]:
```
best.plot_confusion_matrix()
```



[21]:
```
worst.plot_confusion_matrix()
```



Let's see the words the best subject did't get correct.

[22]:
```
for word in best.words_wrong:
    print(f"{word.word:>8} {'('+word.language+')':>12} - {word.meaning}")
```

```
   vogel    (albanian) - small
    gede  (indonesian) - big
  vocerr    (albanian) - small
    tobi      (yoruba) - big
hen xiao    (mandarin) - small
     wei    (mandarin) - small
```

And the words the worst subject got correct.

[23]:
```
for word in worst.words_correct:
    print(f"{word.word:>11} {'('+word.language+')':>12} - {word.meaning}")
```

```
      vogel    (albanian) - small
    raksasa  (indonesian) - big
      besar  (indonesian) - big
         xi    (mandarin) - small
    pang da    (mandarin) - big
     ghanda    (gujarati) - big
       kort       (dutch) - small
```

```
      vigan   (albanian) - big
      kutti      (tamil) - small
        wei   (mandarin) - small
    niraiya      (tamil) - big
ko de ha da     (korean) - big
     tintin     (yoruba) - small
 koskocaman    (turkish) - big
```

There's no a clear pattern in these answers.

2.2.5 Vowel subject

Let's try applying the technique to the distribution of vowels by creating a subject who answers "small" when *I* predominated and answers "big" predominantly for the vowel *A*.

[24]:
```python
from model import Experiment

vowel_subject = Subject(id_="vowel")

for word in words_experiment:

    n_a = word.vowels_count['a']
    n_i = word.vowels_count['i']

    answer = Meaning.SMALL if n_i > n_a else Meaning.BIG

    vowel_subject.add_answer(word_id=word.id_, answer=answer)

vowel_experiment = Experiment(subject=vowel_subject, words=words_experiment)
```

The score of this subject corresponds to guessing 3 words out of 4, and it **performs better than the average of the subjects**. Specifically, it ties with *subject 22* and loses to *subject 12* but beats all the others.

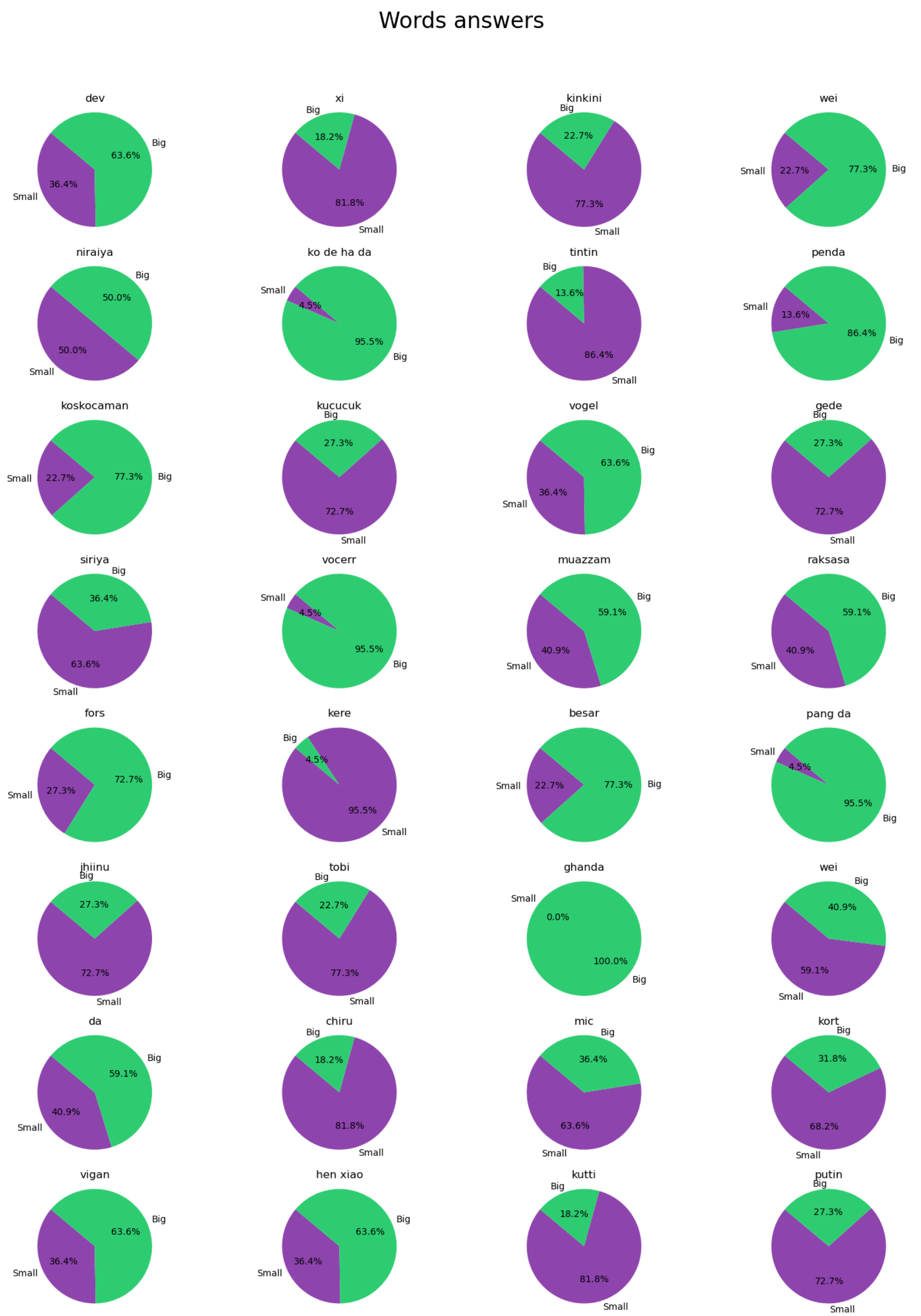
[25]:
```python
print(f"Vowel subject score: {vowel_experiment.score}")
```

```
Vowel subject score: 0.75
```

2.2.6 Word answers

We are now interested in the words and the **distribution between the two classes** (without considering the true label at the moment).

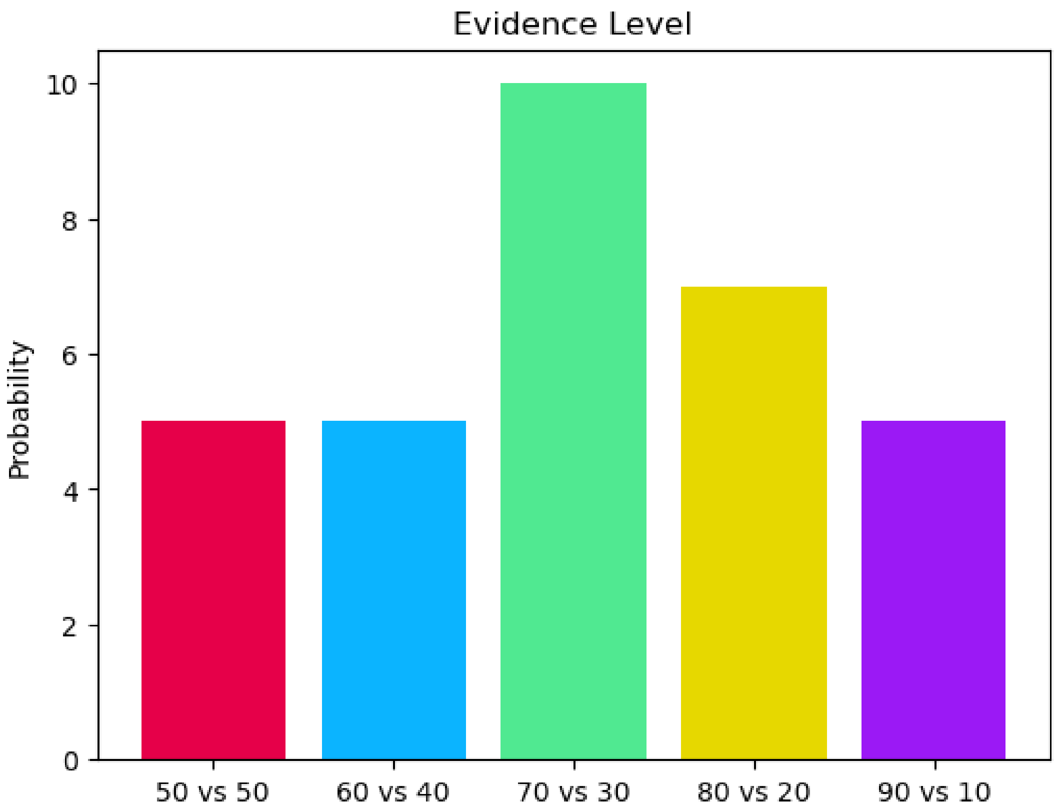
[26]:
```python
experiments.plot_words_answers(n_row=8)
```



The result is *highly variable**, so let's analyze the level of proportionality of responses using five different bins based on the** degree of imbalance**. Bins are more or less balanced, but the fact that there are many cases where responses are unbalanced suggests that** subjects

may share a common pattern** that pushes the answer towards one of the two classes.

[27]:
```
experiments.plot_evidence()
```



Let's inspect words leading to ties and ones leading to a strong imbalance.

[28]:
```
print("No evidence")

for word in experiments.get_evidence()["None"]:
    print(f" - {word.word:>11} {'('+word.language+')':<12} - {str(word.meaning).
 ↪capitalize():>5}")

print()
print("Strong evidence")

for word in experiments.get_evidence()["Strong"]:
    print(f" - {word.word:>11} {'('+word.language+')':<12} - {str(word.meaning).
 ↪capitalize():>5}")
```

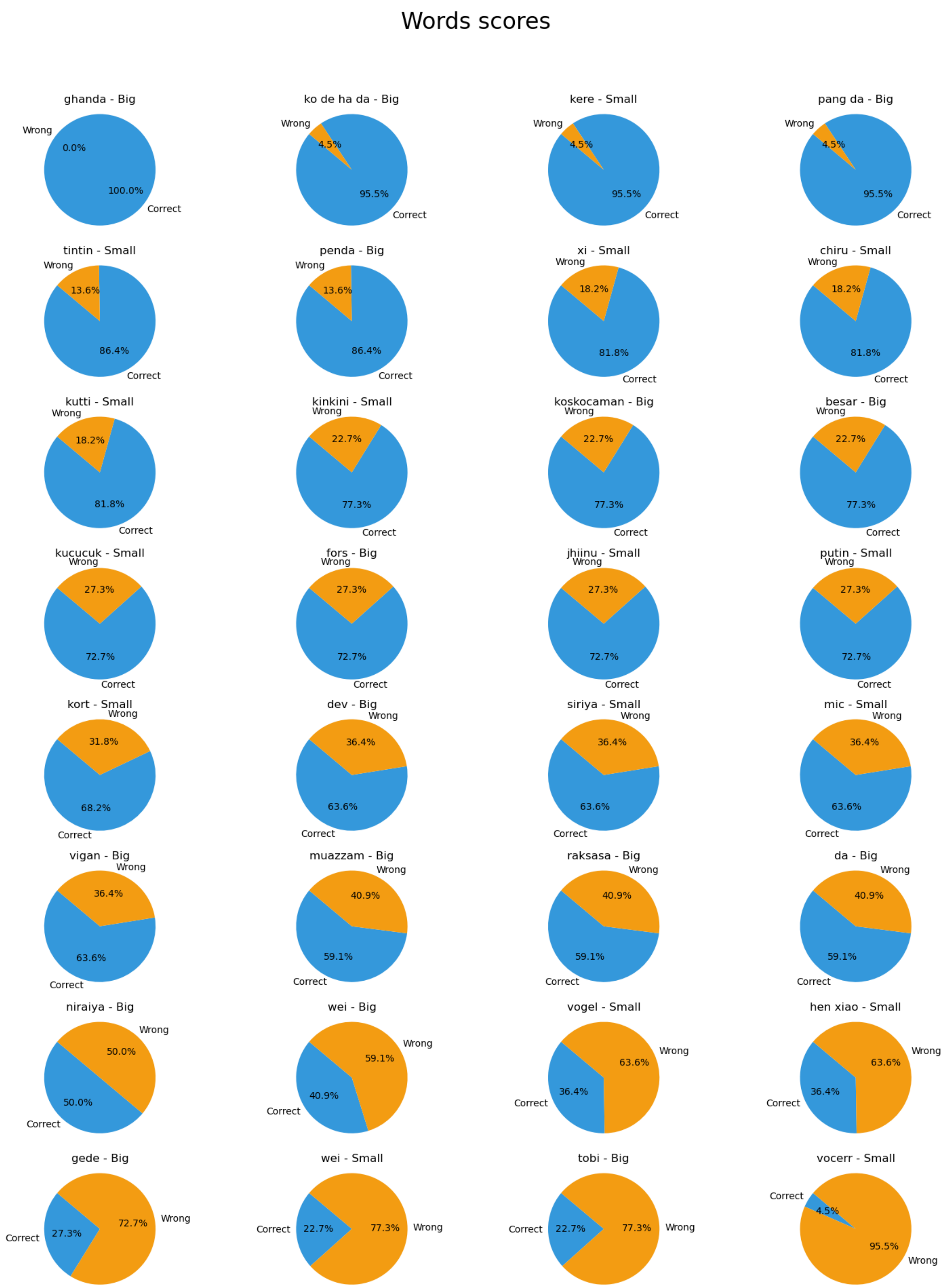
```
No evidence
 -     niraiya (tamil)      -   Big
 -     muazzam (turkish)    -   Big
```

```
 -     raksasa (indonesian) -   Big
 -         wei (mandarin)   -   Big
 -          da (mandarin)   -   Big

Strong evidence
 - ko de ha da (korean)     -   Big
 -      vocerr (albanian)   - Small
 -        kere (yoruba)     - Small
 -     pang da (mandarin)   -   Big
 -      ghanda (gujarati)   -   Big
```

2.2.7 Word scores

We take know into account words answer with respect to the true classes. The results are **fairly balanced**, there emerge both scenarios where the **vast majority of subjects** was able to **guess the correct label** and cases in which **most of them got wrong**.

[29]:
```
experiments.plot_words_scores(n_row=8, sort=True)
```



Let's print the same type of information including information about the language.

[30]:
```python
word_scores = experiments.get_words_results(sort=True)
```

```
for word_id, correct in word_scores:
    word = words_experiment[word_id]
    print(f" - {word.word:>11} {'('+word.language+')':<12} - {str(word.meaning).
↪capitalize():>5}: [{correct:>2}]/[{len(experiments)}]")
```

```
-      ghanda (gujarati)   -   Big: [22]/[22]
- ko de ha da (korean)     -   Big: [21]/[22]
-        kere (yoruba)     - Small: [21]/[22]
-     pang da (mandarin)   -   Big: [21]/[22]
-      tintin (yoruba)     - Small: [19]/[22]
-       penda (gujarati)   -   Big: [19]/[22]
-          xi (mandarin)   - Small: [18]/[22]
-       chiru (tamil)      - Small: [18]/[22]
-       kutti (tamil)      - Small: [18]/[22]
-     kinkini (yoruba)     - Small: [17]/[22]
-  koskocaman (turkish)    -   Big: [17]/[22]
-       besar (indonesian) -   Big: [17]/[22]
-     kucucuk (turkish)    - Small: [16]/[22]
-        fors (dutch)      -   Big: [16]/[22]
-      jhiinu (gujarati)   - Small: [16]/[22]
-       putin (romanian)   - Small: [16]/[22]
-        kort (dutch)      - Small: [15]/[22]
-         dev (turkish)    -   Big: [14]/[22]
-      siriya (tamil)      - Small: [14]/[22]
-         mic (romanian)   - Small: [14]/[22]
-       vigan (albanian)   -   Big: [14]/[22]
-     muazzam (turkish)    -   Big: [13]/[22]
-     raksasa (indonesian) -   Big: [13]/[22]
-          da (mandarin)   -   Big: [13]/[22]
-     niraiya (tamil)      -   Big: [11]/[22]
-         wei (mandarin)   -   Big: [ 9]/[22]
-       vogel (albanian)   - Small: [ 8]/[22]
-    hen xiao (mandarin)   - Small: [ 8]/[22]
-        gede (indonesian) -   Big: [ 6]/[22]
-         wei (mandarin)   - Small: [ 5]/[22]
-        tobi (yoruba)     -   Big: [ 5]/[22]
-      vocerr (albanian)   - Small: [ 1]/[22]
```

2.2.8 3. Neural network

2.2.9 Disclimer

The process of selecting and training the neural network doesn't follow the usual pipeline. There is no hyperparameter tuning, and there's no evaluation of a validation loss to stop the training process, likely leading to overfitting.

This is because the ultimate goal is not to train a high-performing network but to analyze its behavior on a task analogous to that given to human subjects.

2.2.10 Training

The previous analyses suggest that subjects are capable of perceiving a **common pattern** within the word that may be **correlated with the type of meaning**. It can be assumed that this ability stems from being exposed in everyday life to words that refer to the two classes. To simulate this phenomenon, let's **train a neural network** that, given a **word as input**, outputs the **probability that it refers to a large or small object**.

Since the vast majority of subjects are native **Italian** speakers who also know **English** well, let's use words that refer to these two languages for the training of the neural network.

Words are listed in `small.txt` and `big.txt`.

[31]:
```
from io_ import TXTLoader
from settings import SMALL_WORDS_FILE, BIG_WORDS_FILE

small_words_tokens = TXTLoader(file_path=SMALL_WORDS_FILE).load()
big_words_tokens   = TXTLoader(file_path=  BIG_WORDS_FILE).load()
```

Let's see some example.

[32]:
```
import random

print("Small words")
for small_word in random.sample(small_words_tokens, 10):
    print(f"- {small_word}")

print()
print("Big words")
for big_word in random.sample(big_words_tokens, 10):
    print(f"- {big_word}")
```

```
Small words
- stuzzicadenti
- ovatta
- fichetto
- pallottola
- tessuto
- grano di riso
- medaglia
- conchiglia
- lapis
- segnaposto

Big words
- municipality
- asteroid
- cascata
- fiume
- casolare
```

```
- famiglia
- idrovolante
- carousel
- pianeta
- asciugatrice
```

Let's create the collection.

[33]:
```python
words_training = Words()

for i, small_word in enumerate(small_words_tokens, start=1):

    new_word = Word(
        id_=f"sw{i}",
        word=small_word,
        language="it/eng",
        meaning=Meaning.from_string('small'),
        is_sound_symbolic=False
    )

    words_training.add_word(word=new_word)

for i, big_word in enumerate(big_words_tokens, start=1):

    new_word = Word(
        id_=f"bw{i}",
        word=big_word,
        language="it/eng",
        meaning=Meaning.from_string('big'),
        is_sound_symbolic=False
    )

    words_training.add_word(word=new_word)
```

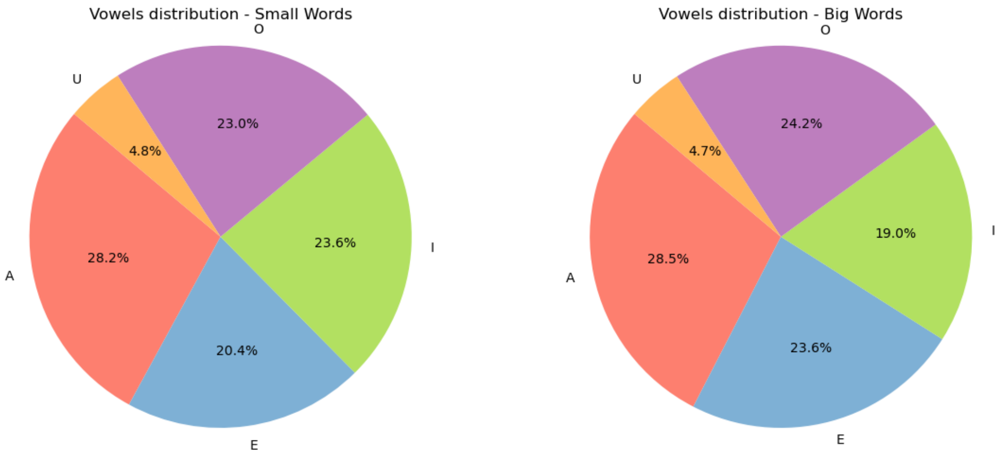
[34]:
```python
words_training
```

[34]:
```
Words[count: 725, small: 311, big: 414]
```

The clear distribution of vowels doesn't repeat as with the experiment words.

[35]:
```python
words_training_small, words_training_big = words_training.get_meaning_split()

fig, axes = plt.subplots(1, 2, figsize=(12, 5))
words_training_small.plot_vowels_distr(ax=axes[0], title="Vowels distribution -␣
↪Small Words")
words_training_big.  plot_vowels_distr(ax=axes[1], title="Vowels distribution -␣
↪Big Words"  )
```

```
plt.tight_layout()
plt.show()
```



We will use a type of **recurrent neural network** (RNN) that operates at the **character level**. This neural network is useful for capturing interdependencies in word construction (it iss, for instance, an excellent use case for the classification of surnames in various languages).

We need to create our own alphabet.

[36]:
```
from network import Alphabet

alphabet = Alphabet()

for word in words_training:
    alphabet.add_word(word=word.word)

alphabet
```

[36]:
```
Alphabet[size: 31]
```

The network has an hyperparameter, the **hidden-size** which we fix to 50.

[37]:
```
from network import MeaningClassifier

HIDDEN_SIZE = 50

model = MeaningClassifier(
    alphabet=alphabet,
    hidden_size=HIDDEN_SIZE,
)

model
```

[37]: MeaningClassifier(
  (input_to_hidden): Embedding(31, 25)
  (hidden_to_hidden): Linear(in_features=50, out_features=25, bias=True)
  (hidden_to_output): Linear(in_features=50, out_features=2, bias=True)
  (softmax): LogSoftmax(dim=1)
)

We create training configurations.

[38]:
```
from settings import DEVICE
from network import ModelConfig

config = ModelConfig(
    epochs=10000,
    epochs_log=150,
    lr=0.001,
    device=DEVICE
)

config
```

[38]: ModelConfig[epochs: 10000; epochs-log: 150; lr: 0.001; device: cpu]

Let's train the network.

[39]:
```
from network import Trainer

trainer = Trainer(
    words=words_training,
    model=model,
    config=config
)
```

[40]:
```
%time
inference = trainer.train()
```

```
CPU times: total: 0 ns
Wall time: 0 ns
Epoch: 150 (1.5%)
  Loss:        0.42324262857437134
  Word:        lavastoviglie
  Meaning:     big
  Prob. Small: 0.34508028626441956
  Prob. Big:   0.6549197435379028
Epoch: 300 (3.0%)
  Loss:        0.36931920051574707
  Word:        match
  Meaning:     small
  Prob. Small: 0.6912047266960144
```

```
  Prob. Big:   0.3087952435016632
Epoch: 450 (4.5%)
  Loss:        0.79610675573349
  Word:        cerniera
  Meaning:     small
  Prob. Small: 0.4510817229747772
  Prob. Big:   0.5489183068275452
Epoch: 600 (6.0%)
  Loss:        0.4740540683269501
  Word:        associazione
  Meaning:     big
  Prob. Small: 0.3775264024734497
  Prob. Big:   0.6224735975265503
Epoch: 750 (7.5%)
  Loss:        0.5602753758430481
  Word:        globulo
  Meaning:     small
  Prob. Small: 0.5710517764091492
  Prob. Big:   0.42894822359085083
Epoch: 900 (9.0%)
  Loss:        0.2746375799179077
  Word:        shoe store
  Meaning:     big
  Prob. Small: 0.240152508020401
  Prob. Big:   0.7598474621772766
Epoch: 1050 (10.5%)
  Loss:        0.5612357258796692
  Word:        pianeta
  Meaning:     big
  Prob. Small: 0.42949631810188293
  Prob. Big:   0.5705036520957947
Epoch: 1200 (12.0%)
  Loss:        0.5255717635154724
  Word:        butterfly
  Meaning:     big
  Prob. Small: 0.4087827801704407
  Prob. Big:   0.5912172198295593
Epoch: 1350 (13.5%)
  Loss:        0.552911639213562
  Word:        strada
  Meaning:     big
  Prob. Small: 0.4247276484966278
  Prob. Big:   0.5752723813056946
Epoch: 1500 (15.0%)
  Loss:        0.7516462206840515
  Word:        piazza
  Meaning:     big
  Prob. Small: 0.5284104347229004
```

```
  Prob. Big:   0.4715895354747772
Epoch: 1650 (16.5%)
  Loss:        0.44099220633506775
  Word:        boccettina
  Meaning:     small
  Prob. Small: 0.6433977484703064
  Prob. Big:   0.3566023111343384
Epoch: 1800 (18.0%)
  Loss:        0.823906660079956
  Word:        bookmark
  Meaning:     small
  Prob. Small: 0.4387143850326538
  Prob. Big:   0.5612856149673462
Epoch: 1950 (19.5%)
  Loss:        0.6950684189796448
  Word:        dwelling
  Meaning:     big
  Prob. Small: 0.5009597539901733
  Prob. Big:   0.49904030561447144
Epoch: 2100 (21.0%)
  Loss:        0.6232422590255737
  Word:        rainbow
  Meaning:     big
  Prob. Small: 0.46379685401916504
  Prob. Big:   0.5362030863761902
Epoch: 2250 (22.5%)
  Loss:        0.6276389360427856
  Word:        rock
  Meaning:     small
  Prob. Small: 0.5338507890701294
  Prob. Big:   0.466149240732193
Epoch: 2400 (24.0%)
  Loss:        1.5882797241210938
  Word:        badge
  Meaning:     small
  Prob. Small: 0.20427674055099487
  Prob. Big:   0.7957231998443604
Epoch: 2550 (25.5%)
  Loss:        0.4091603457927704
  Word:        tappettino
  Meaning:     small
  Prob. Small: 0.6642076969146729
  Prob. Big:   0.33579227328300476
Epoch: 2700 (27.0%)
  Loss:        0.3103548586368561
  Word:        pendant
  Meaning:     small
  Prob. Small: 0.7331867218017578
```

```
  Prob. Big:   0.2668132781982422
Epoch: 2850 (28.5%)
  Loss:        0.35583263635635376
  Word:        perfumery
  Meaning:     big
  Prob. Small: 0.2994101345539093
  Prob. Big:   0.7005898356437683
Epoch: 3000 (30.0%)
  Loss:        0.21687883138656616
  Word:        ventilatore
  Meaning:     big
  Prob. Small: 0.1949724704027176
  Prob. Big:   0.8050274848937988
Epoch: 3150 (31.5%)
  Loss:        0.42235267162323
  Word:        confederation
  Meaning:     big
  Prob. Small: 0.3444971740245819
  Prob. Big:   0.6555028557777405
Epoch: 3300 (33.0%)
  Loss:        0.5074808597564697
  Word:        railway
  Meaning:     big
  Prob. Small: 0.39798980951309204
  Prob. Big:   0.6020102500915527
Epoch: 3450 (34.5%)
  Loss:        0.6324858665466309
  Word:        city
  Meaning:     big
  Prob. Small: 0.4687305688858032
  Prob. Big:   0.5312694907188416
Epoch: 3600 (36.0%)
  Loss:        0.5507564544677734
  Word:        cuffietta
  Meaning:     small
  Prob. Small: 0.5765135288238525
  Prob. Big:   0.42348647117614746
Epoch: 3750 (37.5%)
  Loss:        1.0608083009719849
  Word:        pebble
  Meaning:     small
  Prob. Small: 0.346175879240036
  Prob. Big:   0.6538241505622864
Epoch: 3900 (39.0%)
  Loss:        0.3825025260448456
  Word:        nazione
  Meaning:     big
  Prob. Small: 0.3178478181362152
```

```
  Prob. Big:   0.6821521520614624
Epoch: 4050 (40.5%)
  Loss:        0.48110735416412354
  Word:        seal
  Meaning:     small
  Prob. Small: 0.6180985569953918
  Prob. Big:   0.38190144300460815
Epoch: 4200 (42.0%)
  Loss:        0.18540357053279877
  Word:        refrigerator
  Meaning:     big
  Prob. Small: 0.1692310869693756
  Prob. Big:   0.8307689428329468
Epoch: 4350 (43.5%)
  Loss:        0.5187988877296448
  Word:        provincia
  Meaning:     big
  Prob. Small: 0.40476492047309875
  Prob. Big:   0.5952350497245789
Epoch: 4500 (45.0%)
  Loss:        0.5448006391525269
  Word:        tavolo
  Meaning:     big
  Prob. Small: 0.42004257440567017
  Prob. Big:   0.5799573659896851
Epoch: 4650 (46.5%)
  Loss:        0.3873543441295624
  Word:        centrino
  Meaning:     small
  Prob. Small: 0.678850531578064
  Prob. Big:   0.3211495280265808
Epoch: 4800 (48.0%)
  Loss:        0.5508548021316528
  Word:        borsetta
  Meaning:     small
  Prob. Small: 0.5764568448066711
  Prob. Big:   0.42354312539100647
Epoch: 4950 (49.5%)
  Loss:        0.21544204652309418
  Word:        cattedrale
  Meaning:     big
  Prob. Small: 0.1938149780035019
  Prob. Big:   0.8061850070953369
Epoch: 5100 (51.0%)
  Loss:        0.3046257495880127
  Word:        piumetta
  Meaning:     small
  Prob. Small: 0.7373992800712585
```

```
  Prob. Big:   0.26260069012641907
Epoch: 5250 (52.5%)
  Loss:        0.2689824104309082
  Word:        paracolpi
  Meaning:     small
  Prob. Small: 0.764156699180603
  Prob. Big:   0.23584336042404175
Epoch: 5400 (54.0%)
  Loss:        1.0764338970184326
  Word:        grotta
  Meaning:     big
  Prob. Small: 0.6591913104057312
  Prob. Big:   0.3408087193965912
Epoch: 5550 (55.5%)
  Loss:        0.23101501166820526
  Word:        tappettino
  Meaning:     small
  Prob. Small: 0.7937275171279907
  Prob. Big:   0.2062724232673645
Epoch: 5700 (57.0%)
  Loss:        0.2945529818534851
  Word:        quadretto
  Meaning:     small
  Prob. Small: 0.7448644638061523
  Prob. Big:   0.2551354765892029
Epoch: 5850 (58.5%)
  Loss:        0.6019206643104553
  Word:        piazza
  Meaning:     big
  Prob. Small: 0.4522414207458496
  Prob. Big:   0.5477585792541504
Epoch: 6000 (60.0%)
  Loss:        1.056868076324463
  Word:        pen
  Meaning:     small
  Prob. Small: 0.34754255414009094
  Prob. Big:   0.6524573564529419
Epoch: 6150 (61.5%)
  Loss:        0.3134448230266571
  Word:        fondazione
  Meaning:     big
  Prob. Small: 0.26907533407211304
  Prob. Big:   0.7309247255325317
Epoch: 6300 (63.0%)
  Loss:        1.5751748085021973
  Word:        giardino
  Meaning:     big
  Prob. Small: 0.7930286526679993
```

```
  Prob. Big:   0.20697136223316193
Epoch: 6450 (64.5%)
  Loss:        0.34175005555152893
  Word:        cristallo
  Meaning:     small
  Prob. Small: 0.7105257511138916
  Prob. Big:   0.28947415947914124
Epoch: 6600 (66.0%)
  Loss:        0.6592358350753784
  Word:        medaglia
  Meaning:     small
  Prob. Small: 0.517246425151825
  Prob. Big:   0.48275354504585266
Epoch: 6750 (67.5%)
  Loss:        0.7210822105407715
  Word:        moschea
  Meaning:     big
  Prob. Small: 0.5137742161750793
  Prob. Big:   0.48622578382492065
Epoch: 6900 (69.0%)
  Loss:        0.1959645003080368
  Word:        gate
  Meaning:     big
  Prob. Small: 0.17795857787132263
  Prob. Big:   0.822041392326355
Epoch: 7050 (70.5%)
  Loss:        0.44067898392677307
  Word:        optics
  Meaning:     big
  Prob. Small: 0.356400728225708
  Prob. Big:   0.643599271774292
Epoch: 7200 (72.0%)
  Loss:        0.32554638385772705
  Word:        dischetto
  Meaning:     small
  Prob. Small: 0.722132682800293
  Prob. Big:   0.27786731719970703
Epoch: 7350 (73.5%)
  Loss:        0.8945173621177673
  Word:        telescopio
  Meaning:     big
  Prob. Small: 0.5911951065063477
  Prob. Big:   0.40880486369132996
Epoch: 7500 (75.0%)
  Loss:        0.09777727723121643
  Word:        kingdom
  Meaning:     big
  Prob. Small: 0.09314918518066406
```

```
  Prob. Big:   0.9068508744239807
Epoch: 7650 (76.5%)
  Loss:        0.3315032720565796
  Word:        macelleria
  Meaning:     big
  Prob. Small: 0.2821561396121979
  Prob. Big:   0.7178438305854797
Epoch: 7800 (78.0%)
  Loss:        0.9163187146186829
  Word:        pietra
  Meaning:     big
  Prob. Small: 0.6000112295150757
  Prob. Big:   0.3999888002872467
Epoch: 7950 (79.5%)
  Loss:        0.6072944402694702
  Word:        rock
  Meaning:     small
  Prob. Small: 0.5448229312896729
  Prob. Big:   0.45517706871032715
Epoch: 8100 (81.0%)
  Loss:        0.35440173745155334
  Word:        galleria
  Meaning:     big
  Prob. Small: 0.2984069287776947
  Prob. Big:   0.7015930414199829
Epoch: 8250 (82.5%)
  Loss:        0.14432436227798462
  Word:        road
  Meaning:     big
  Prob. Small: 0.13439302146434784
  Prob. Big:   0.8656069040298462
Epoch: 8400 (84.0%)
  Loss:        0.3482995927333832
  Word:        freccetta
  Meaning:     small
  Prob. Small: 0.7058873772621155
  Prob. Big:   0.2941126525402069
Epoch: 8550 (85.5%)
  Loss:        0.39653006196022034
  Word:        tavolo
  Meaning:     big
  Prob. Small: 0.3273499608039856
  Prob. Big:   0.6726500391960144
Epoch: 8700 (87.0%)
  Loss:        0.34974709153175354
  Word:        coccinella
  Meaning:     small
  Prob. Small: 0.704866349697113
```

```
  Prob. Big:   0.29513365030288696
Epoch: 8850 (88.5%)
  Loss:        1.5016403198242188
  Word:        keychain
  Meaning:     small
  Prob. Small: 0.22276444733142853
  Prob. Big:   0.7772355675697327
Epoch: 9000 (90.0%)
  Loss:        0.6028662323951721
  Word:        tessuto
  Meaning:     small
  Prob. Small: 0.5472408533096313
  Prob. Big:   0.45275914669036865
Epoch: 9150 (91.5%)
  Loss:        1.0047118663787842
  Word:        baita
  Meaning:     big
  Prob. Small: 0.6338499188423157
  Prob. Big:   0.3661501407623291
Epoch: 9300 (93.0%)
  Loss:        1.052327275276184
  Word:        canottiera
  Meaning:     small
  Prob. Small: 0.3491242527961731
  Prob. Big:   0.6508757472038269
Epoch: 9450 (94.5%)
  Loss:        0.41914689540863037
  Word:        rametto
  Meaning:     small
  Prob. Small: 0.6576075553894043
  Prob. Big:   0.3423924148082733
Epoch: 9600 (96.0%)
  Loss:        0.10071633011102676
  Word:        gate
  Meaning:     big
  Prob. Small: 0.09581048786640167
  Prob. Big:   0.9041894674301147
Epoch: 9750 (97.5%)
  Loss:        0.31484416127204895
  Word:        lavoretto
  Meaning:     small
  Prob. Small: 0.7299026250839233
  Prob. Big:   0.27009740471839905
Epoch: 9900 (99.0%)
  Loss:        0.8715804219245911
  Word:        magnet
  Meaning:     small
  Prob. Small: 0.4182899296283722
```

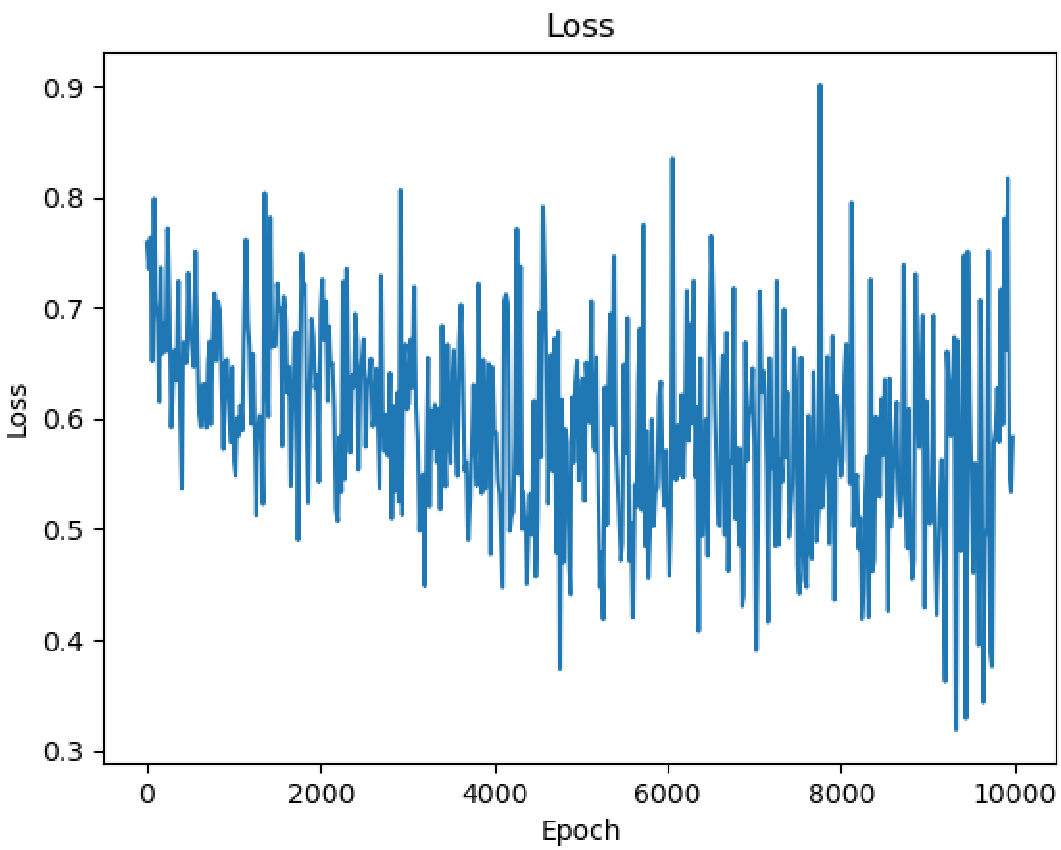
```
Prob. Big:   0.5817100405693054
```

The training loss decreases initially but then stabilizes and begins to oscillate, indicating that a certain learning capacity has been achieved.

2.2.11 Model analysis

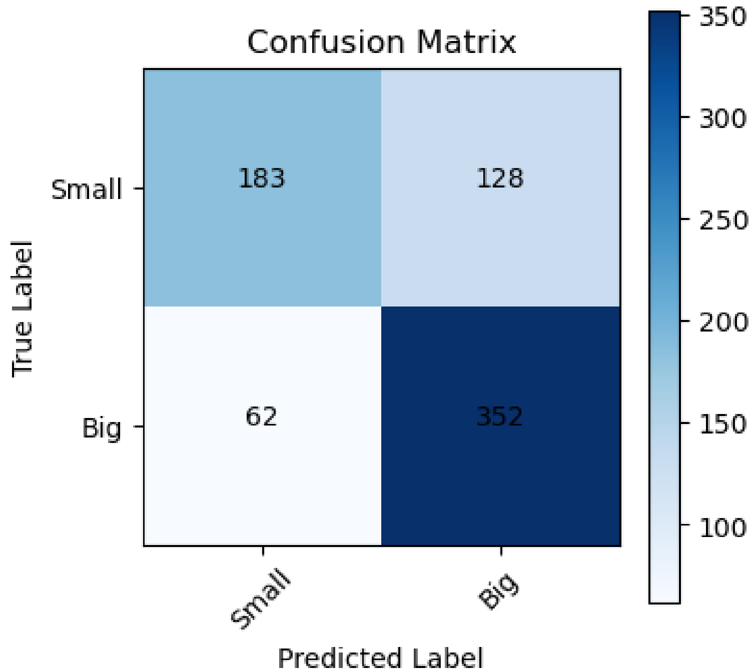
The training loss decreases initially but then stabilizes and begins to oscillate, indicating that a certain learning capacity has been achieved.

[41]:
```
inference.plot_loss(title="Loss")
```



The confusion matrix is satisfactory, but perhaps it's indicative of overfitting. For this reason, we also evaluate the model on a test set that hasn't been seen during training to ensure a more comprehensive assessment.

[42]:
```
inference.plot_confusion_matrix(words=words_training)
```

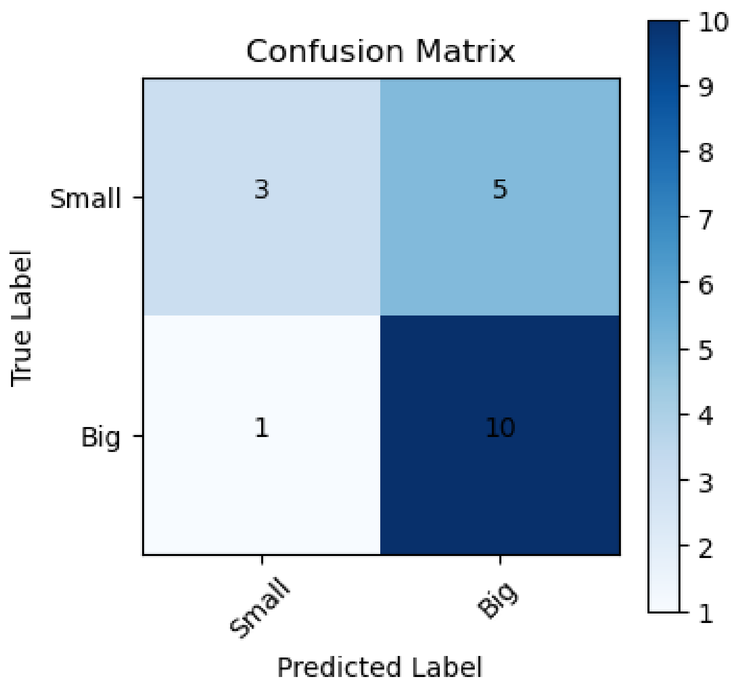


Let's also evaluate predictions on a test set. The model is probably prone to predict "big" class as it's baised on the training set which contains more big words.

[43]:
```
from settings import SMALL_WORDS_FILE_TEST, BIG_WORDS_FILE_TEST

small_words_test_tokens = TXTLoader(file_path=SMALL_WORDS_FILE_TEST).load()
big_words_test_tokens   = TXTLoader(file_path=  BIG_WORDS_FILE_TEST).load()

words_test = Words()

for i, small_word in enumerate(small_words_test_tokens, start=1):

    new_word = Word(
        id_=f"sw{i}_test",
        word=small_word,
        language="it/eng",
        meaning=Meaning.from_string('small'),
        is_sound_symbolic=False
    )

    words_test.add_word(word=new_word)

for i, big_word in enumerate(big_words_test_tokens, start=1):

    new_word = Word(
```

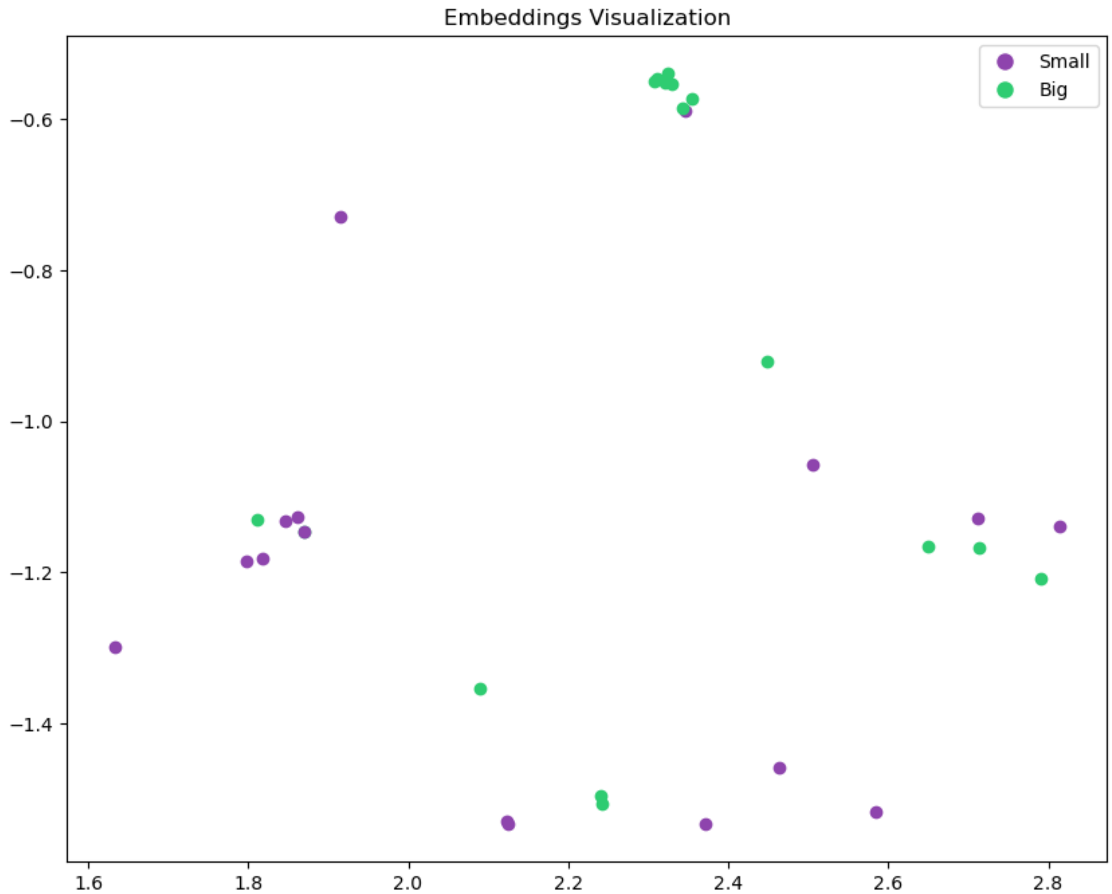
```
        id_=f"bw{i}_test",
        word=big_word,
        language="it/eng",
        meaning=Meaning.from_string('big'),
        is_sound_symbolic=False
    )

    words_test.add_word(word=new_word)
```

[44]:
```
inference.plot_confusion_matrix(words=words_test)
```



Finally, we plot word embeddings in a 2-dimensional space using TSNE dimensionality reduction which is expected to maintain clusters of points when projecting to lower dimension, but no clear pattern appears.

[45]:
```
inference.plot_embedding(words=words_experiment)
```

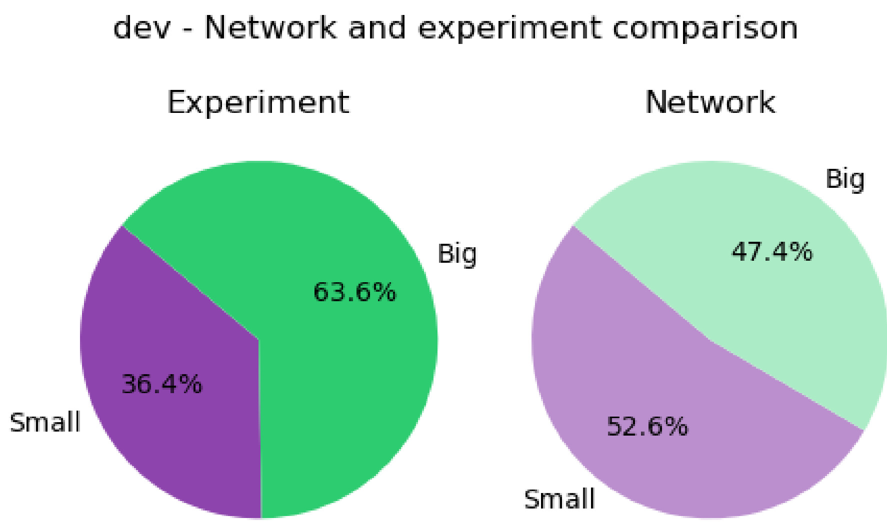


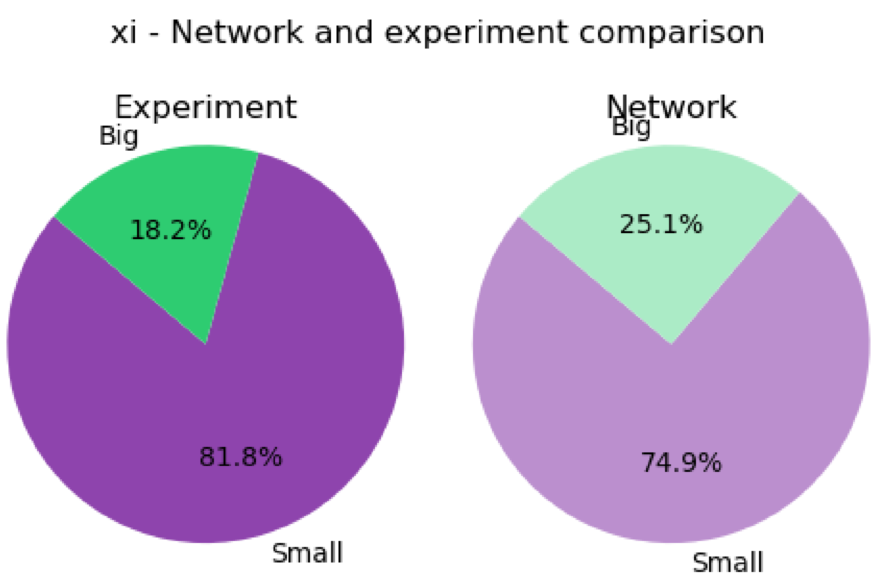
2.2.12 Comparison with Human assessment

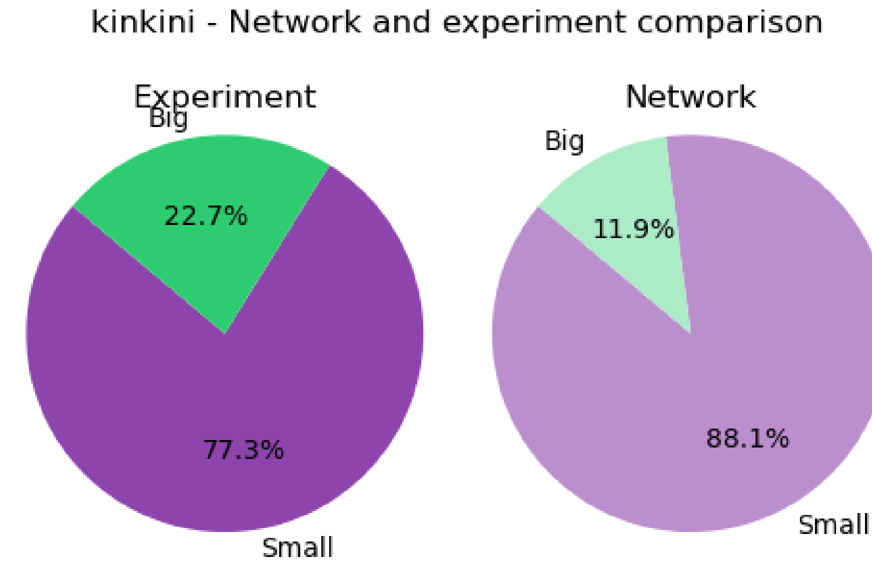
For each word, let's compare pie charts between the neural network and human subjects. It's important to remember that the two have **entirely different interpretations**. The proportion in the human experiment comes from an average across multiple subjects responding to a certain class, while the neural network directly returns the probability distribution for each class using a softmax.

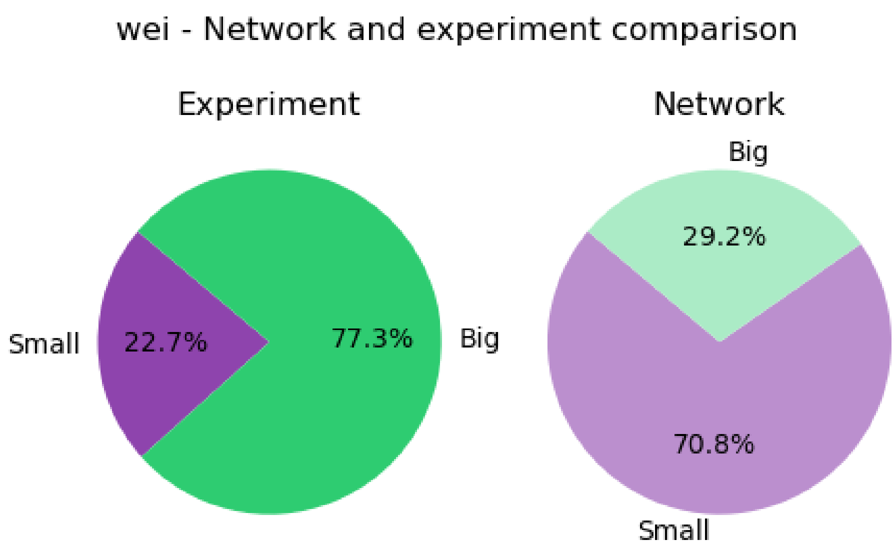
[46]:
```
from settings import SMALL_BIG_COLORS, NETWORK_COLORS
from io_ import pie_plot

for word_id, answer in experiments.get_word_answers().items():

    word = words_experiment[word_id]

    experiment_probs = [count / sum(answer) for count in answer]
    network_probs = inference.predict(word=word.word)

    labels = ["Small", "Big"]
```

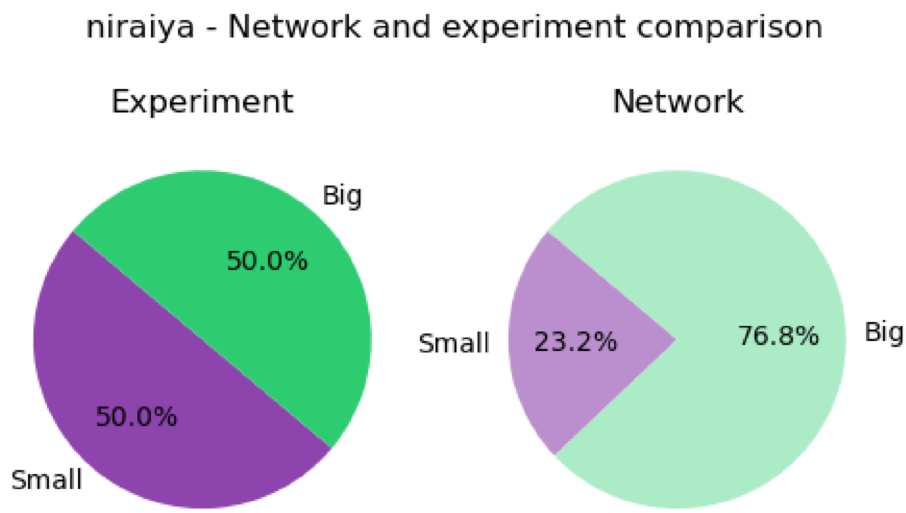
```
    fig, axes = plt.subplots(1, 2, figsize=(5, 3))
    pie_plot(labels=labels, sizes=experiment_probs, colors=SMALL_BIG_COLORS,␣
↪title="Experiment", ax=axes[0])
    pie_plot(labels=labels, sizes=network_probs,    colors=NETWORK_COLORS,␣
↪title="Network", ax=axes[1])

    fig.suptitle(f"{word.word} - Network and experiment comparison")
    fig.tight_layout()
    plt.show()
```

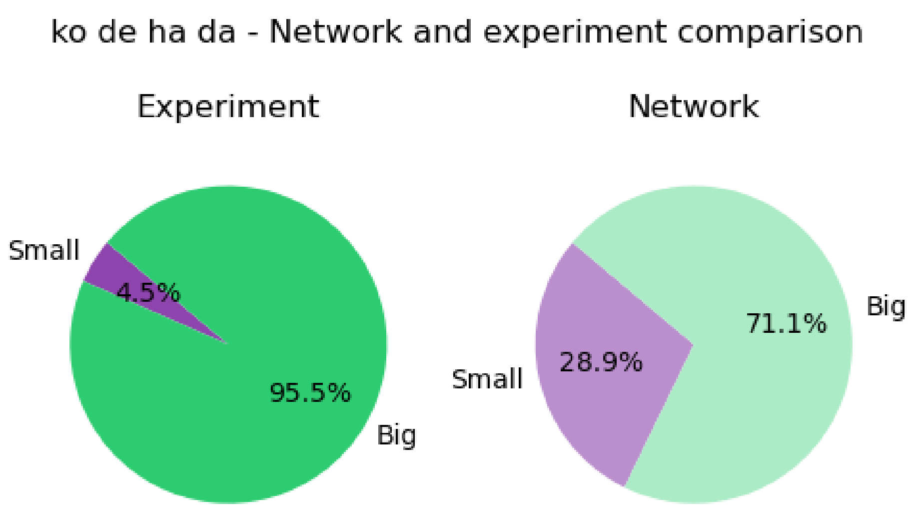


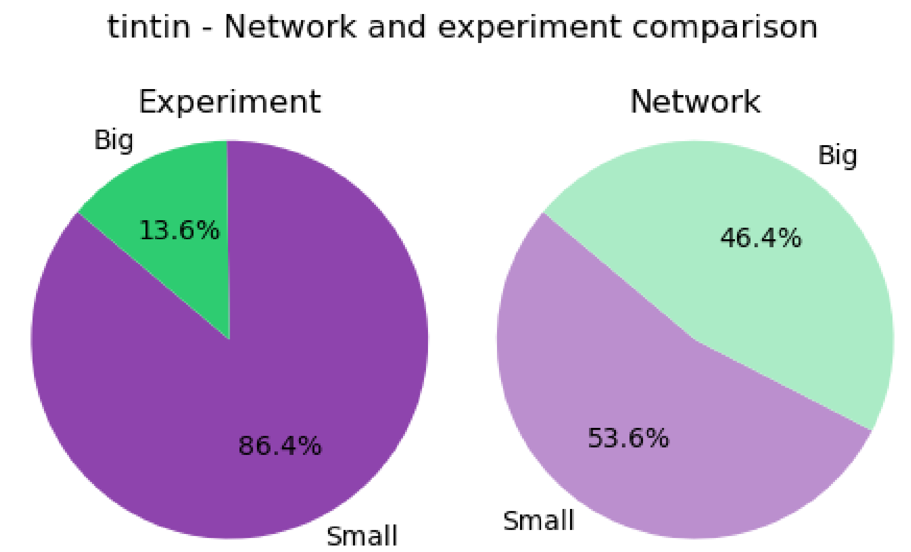


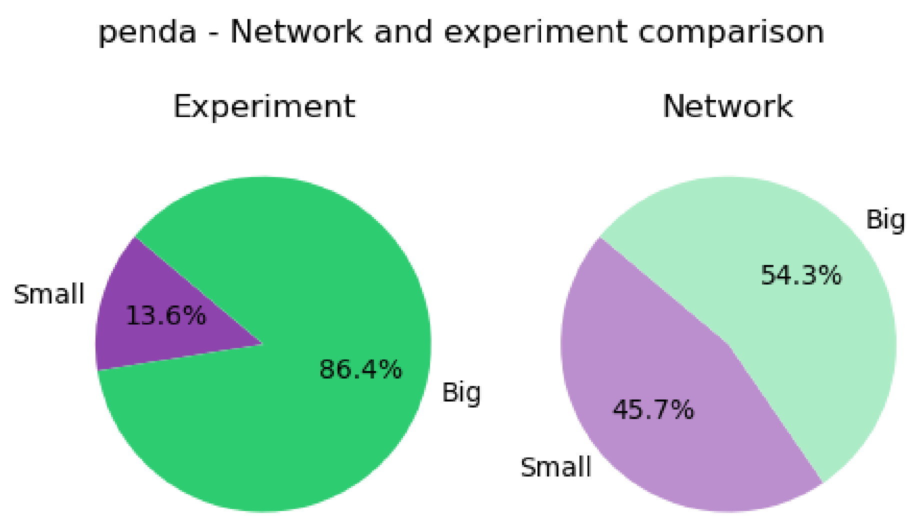

kinkini - Network and experiment comparison
Experiment
Big
22.7%
77.3%
Small
Network
Big
11.9%
88.1%
Small

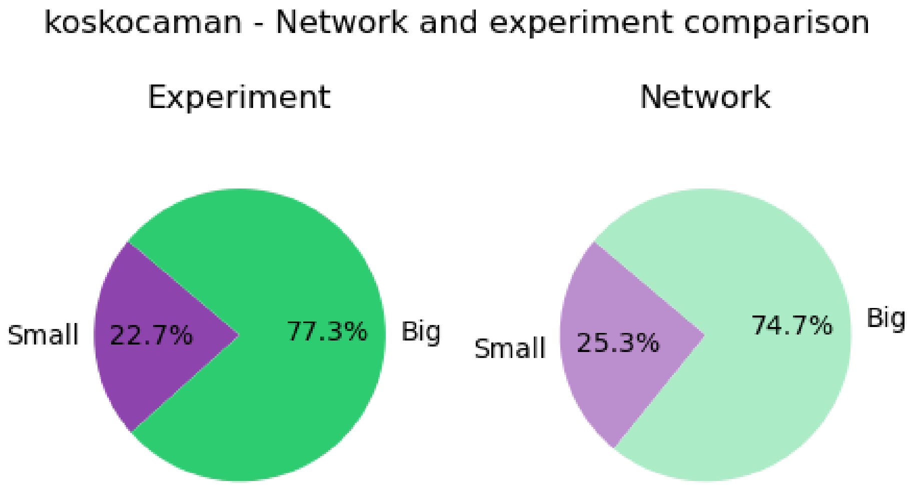

wei - Network and experiment comparison
Experiment
Small
22.7%
77.3%
Big
Network
Big
29.2%
70.8%
Small

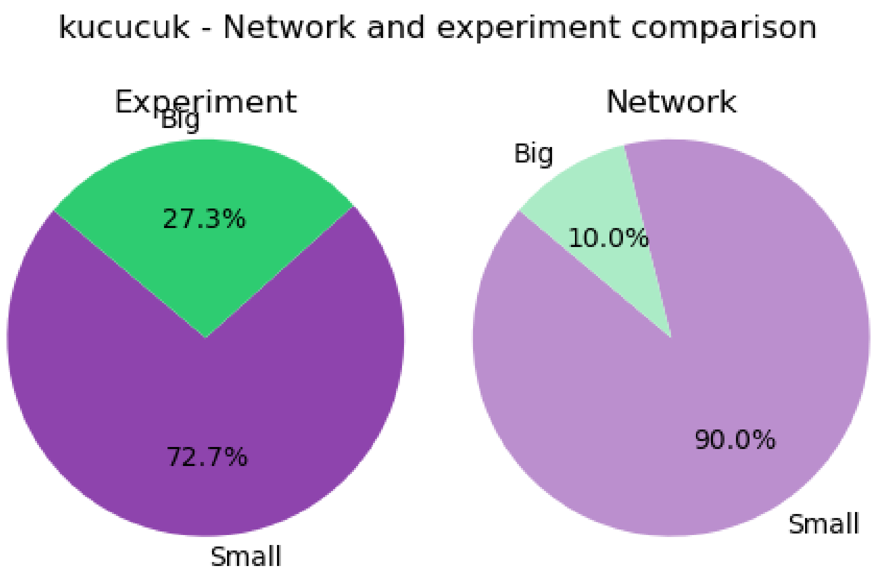

niraiya - Network and experiment comparison
Experiment
Network
Big
50.0%
50.0%
Small
Small
23.2%
76.8%
Big

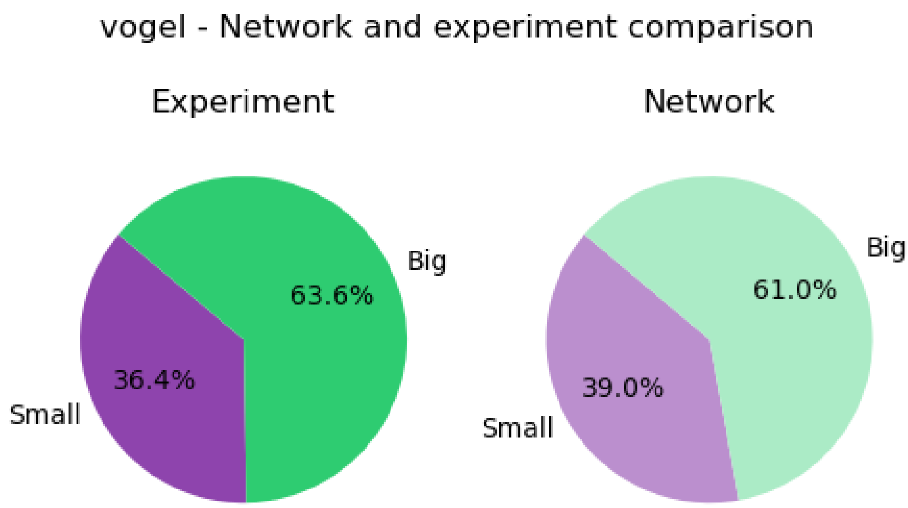

ko de ha da - Network and experiment comparison
Experiment
Network
Small
4.5%
95.5%
Big
Small
28.9%
71.1%
Big

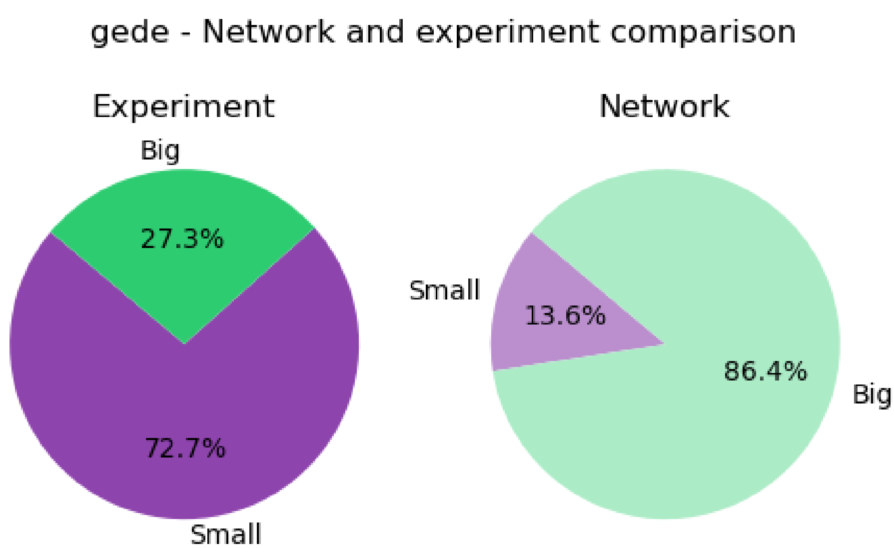

tintin - Network and experiment comparison
Experiment
Network
Big
13.6%
86.4%
Small
Big
46.4%
53.6%
Small

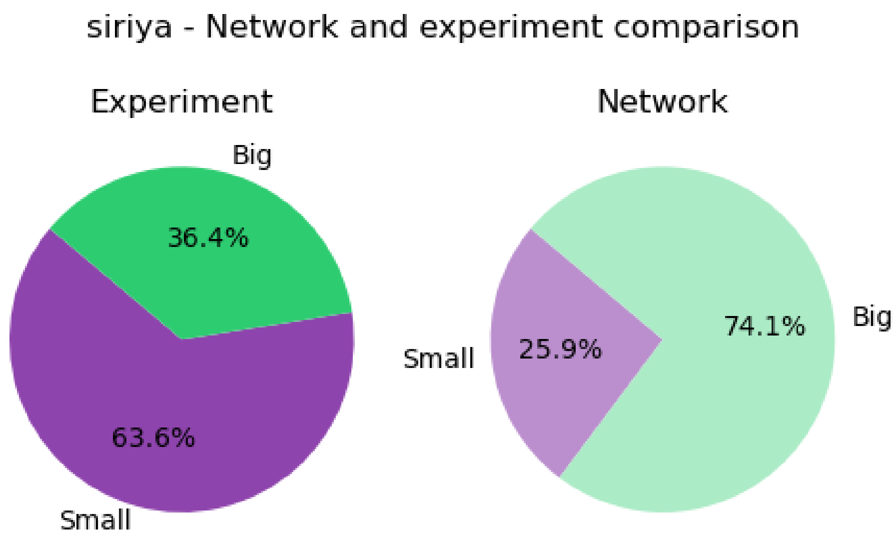

penda - Network and experiment comparison
Experiment
Network
Small
13.6%
86.4%
Big
Big
54.3%
45.7%
Small

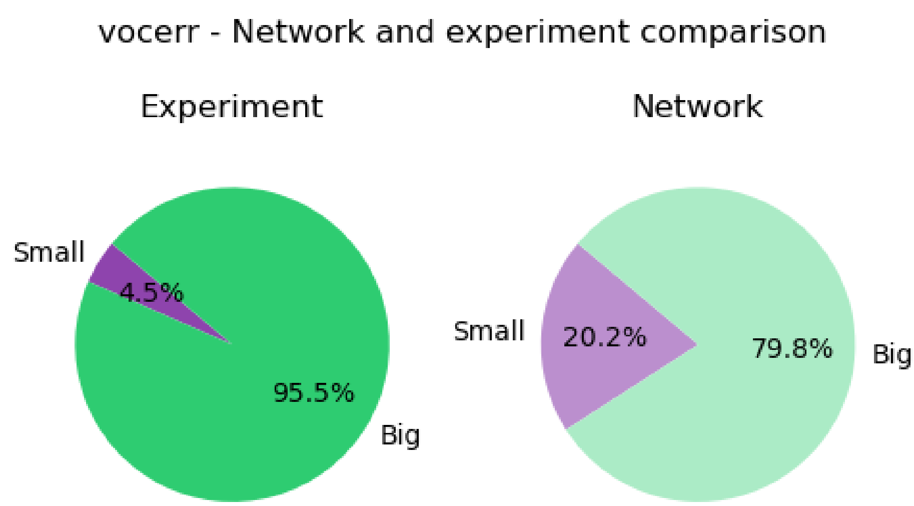

koskocaman - Network and experiment comparison
Experiment
Network
Small
22.7%
77.3%
Big
Small
25.3%
74.7%
Big

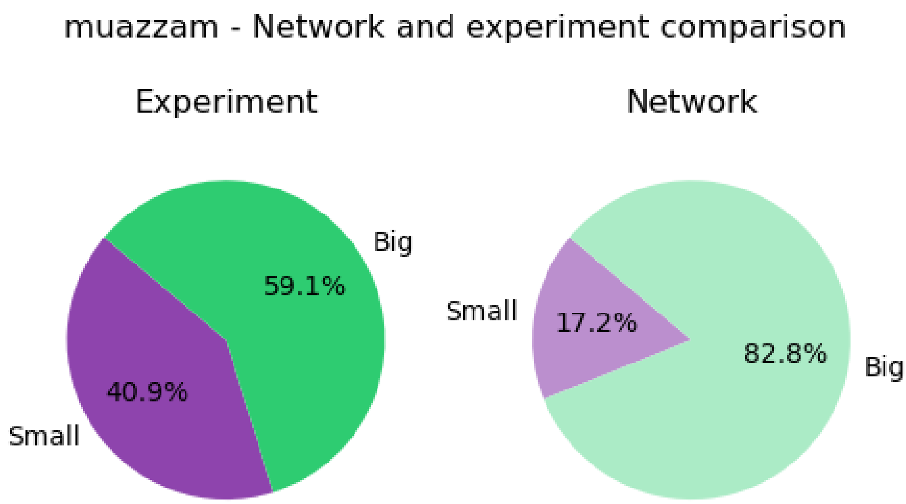

kucucuk - Network and experiment comparison
Experiment
Network
Big
27.3%
72.7%
Small
Big
10.0%
90.0%
Small

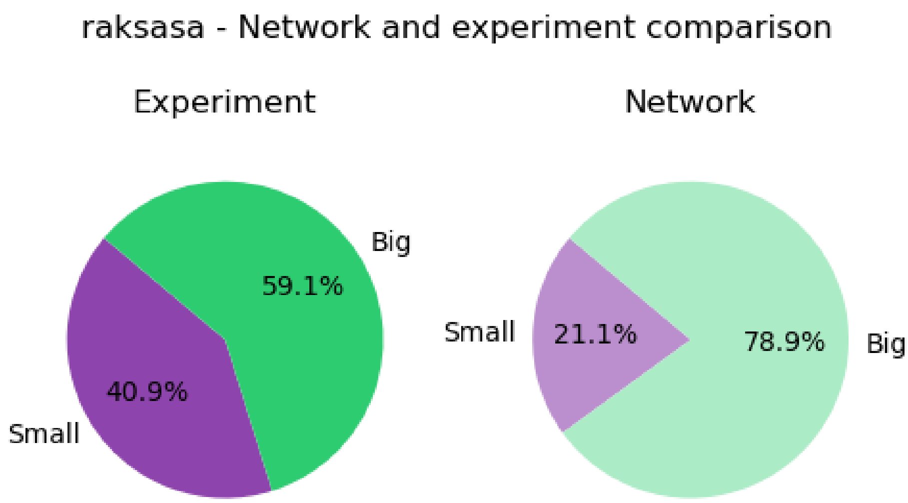

vogel - Network and experiment comparison
Experiment
Big
63.6%
36.4%
Small
Network
Big
61.0%
39.0%
Small

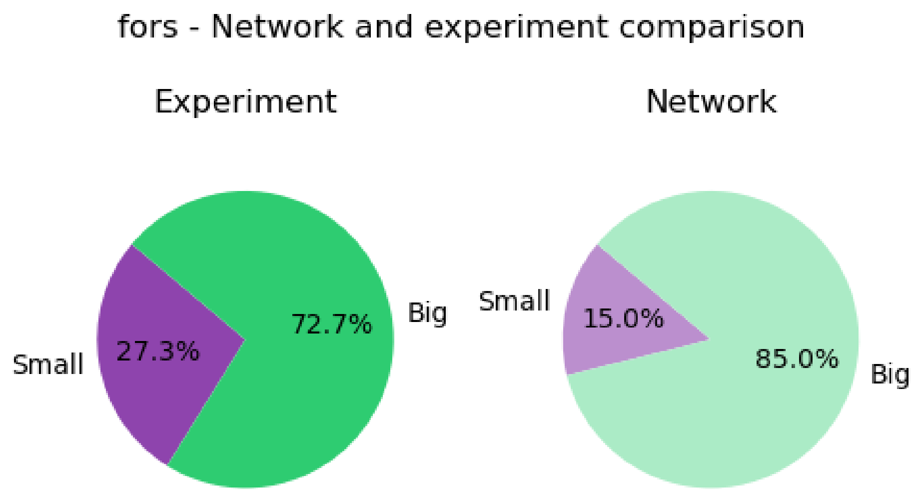

gede - Network and experiment comparison
Experiment
Big
27.3%
72.7%
Small
Network
Small
13.6%
86.4%
Big

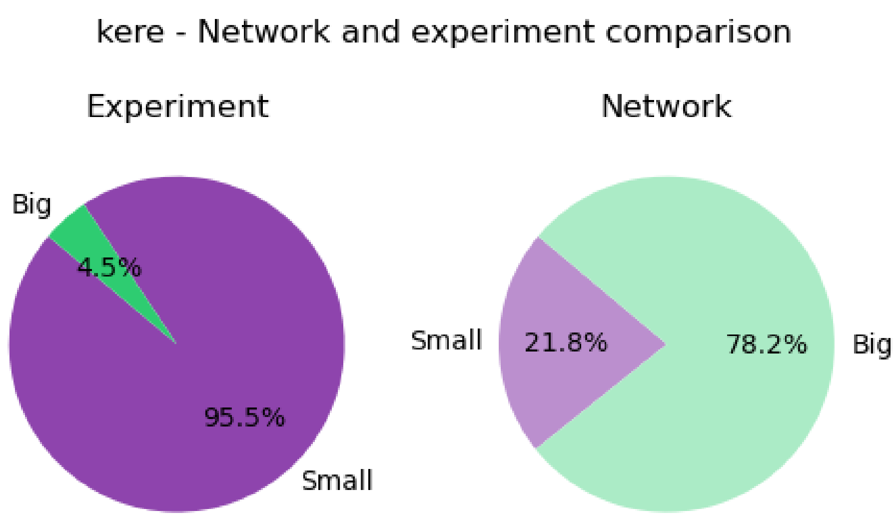

siriya - Network and experiment comparison
Experiment
Big
36.4%
63.6%
Small
Network
Small
25.9%
74.1%
Big

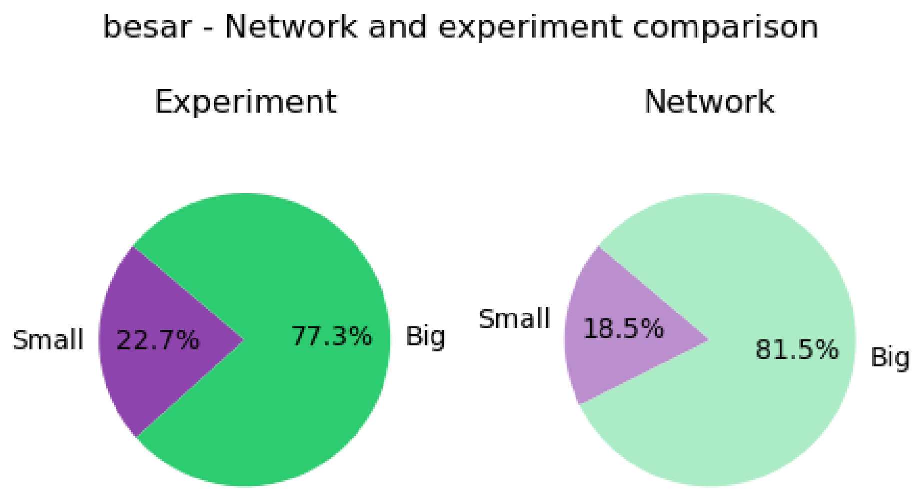

vocerr - Network and experiment comparison
Experiment
Small
4.5%
95.5%
Big
Network
Small
20.2%
79.8%
Big

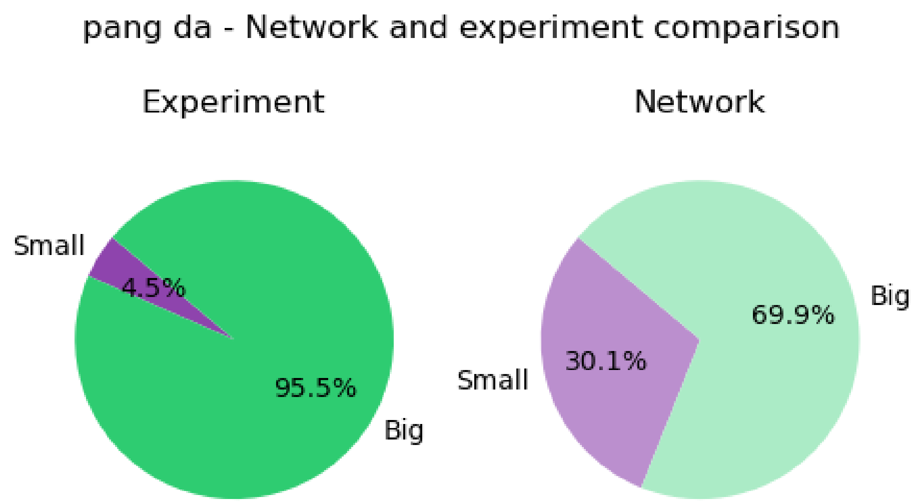

muazzam - Network and experiment comparison
Experiment
Network
Big
59.1%
40.9%
Small
Small
17.2%
82.8%
Big

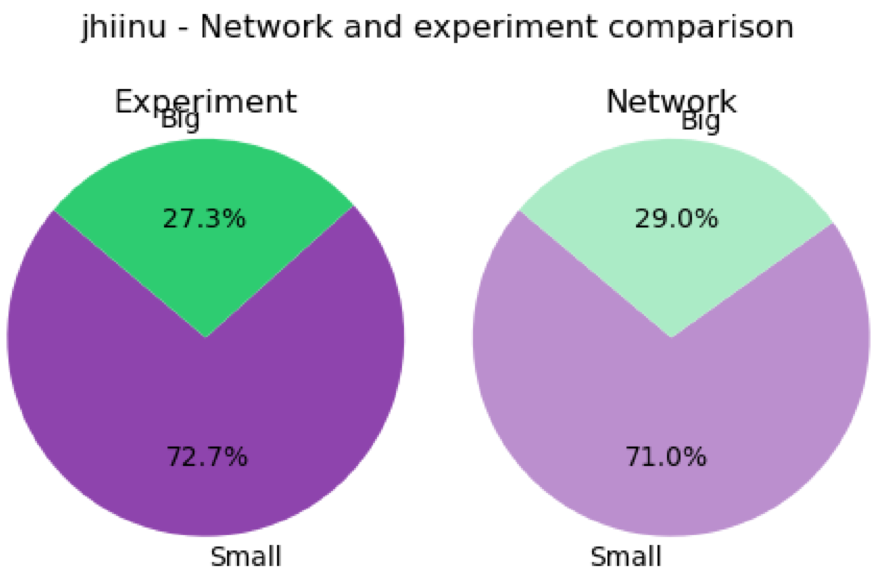

raksasa - Network and experiment comparison
Experiment
Network
Big
59.1%
40.9%
Small
Small
21.1%
78.9%
Big

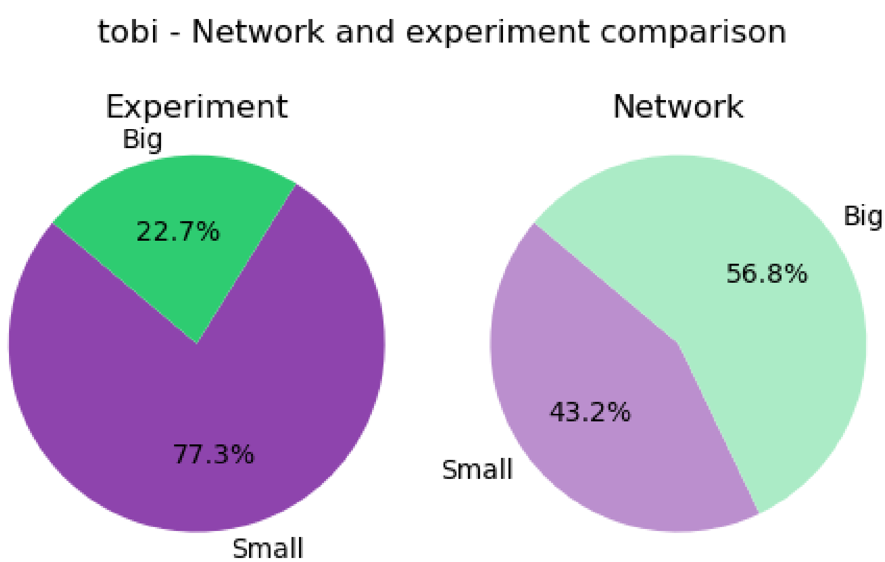

fors - Network and experiment comparison
Experiment
Network
Small
27.3%
72.7%
Big
Small
15.0%
85.0%
Big

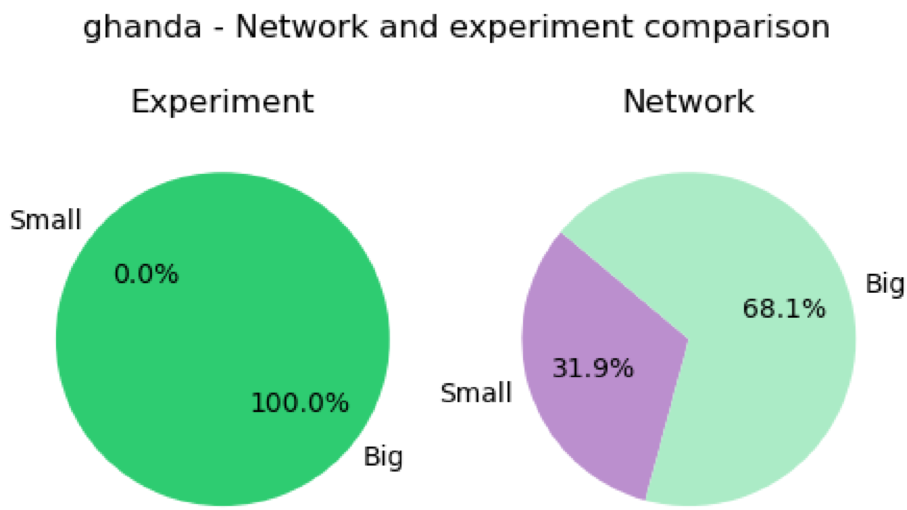

kere - Network and experiment comparison
Experiment
Network
Big
4.5%
95.5%
Small
Small
21.8%
78.2%
Big

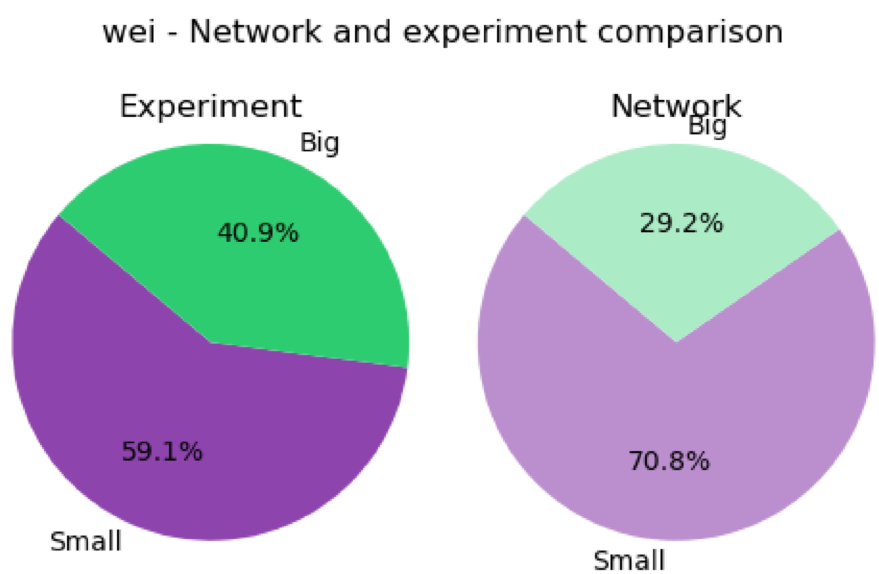

besar - Network and experiment comparison
Experiment
Network
Small
22.7%
77.3%
Big
Small
18.5%
81.5%
Big

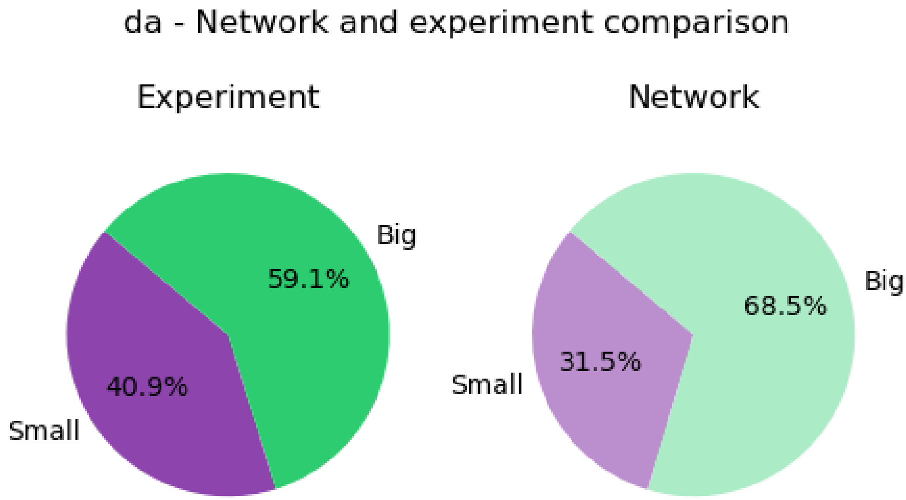

pang da - Network and experiment comparison
Experiment
Network
Small
4.5%
95.5%
Big
Small
30.1%
69.9%
Big

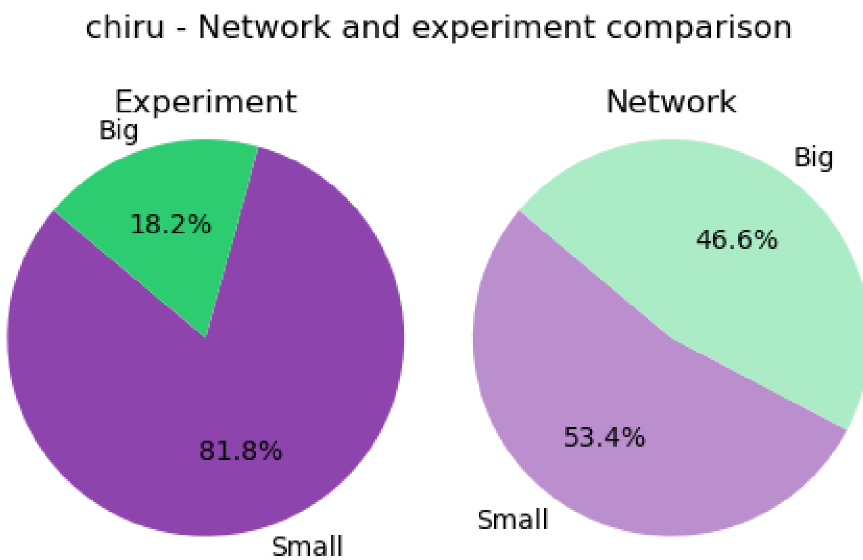

jhiinu - Network and experiment comparison
Experiment
Big
27.3%
72.7%
Small
Network
Big
29.0%
71.0%
Small

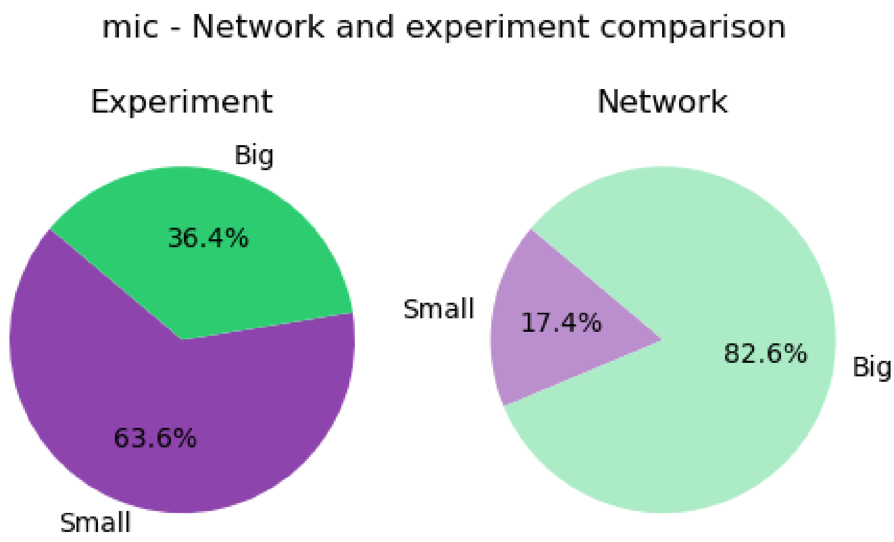

tobi - Network and experiment comparison
Experiment
Big
22.7%
77.3%
Small
Network
Big
56.8%
43.2%
Small

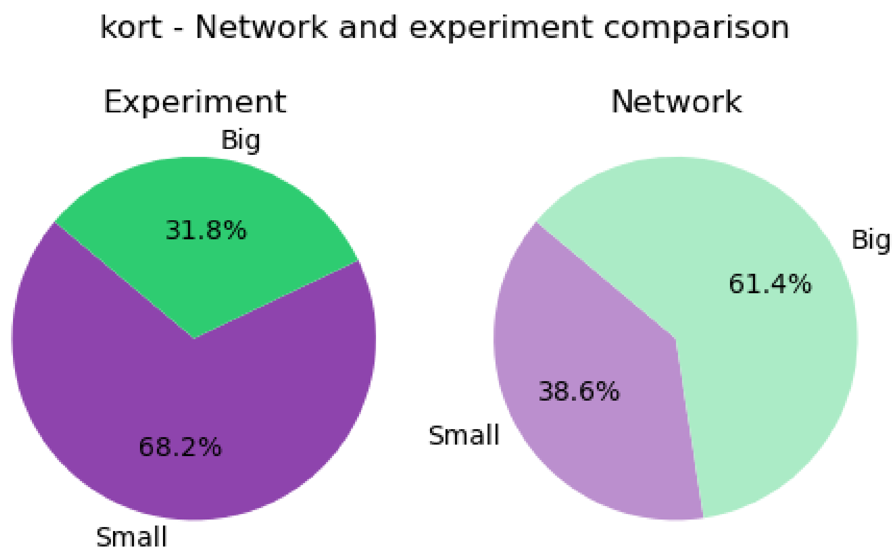

ghanda - Network and experiment comparison
Experiment
Network
Small
0.0%
100.0%
Big
Big
68.1%
31.9%
Small

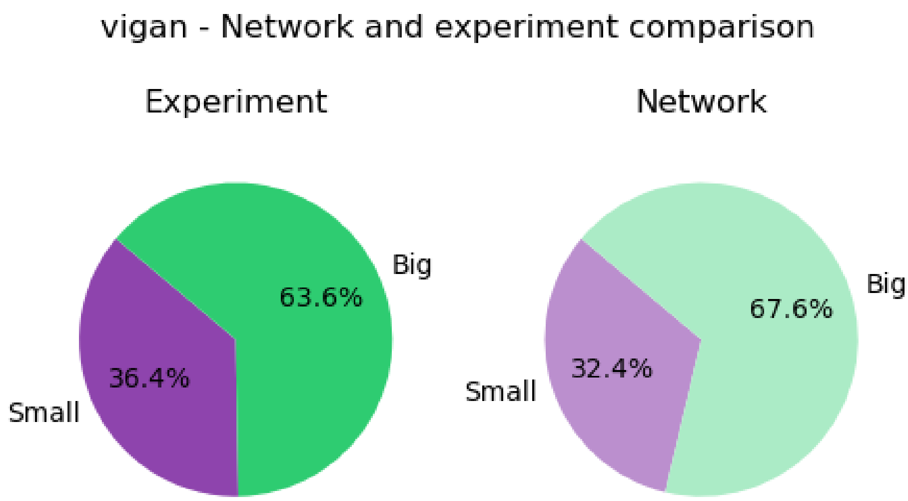

wei - Network and experiment comparison
Experiment
Network
Big
40.9%
59.1%
Small
Big
29.2%
70.8%
Small

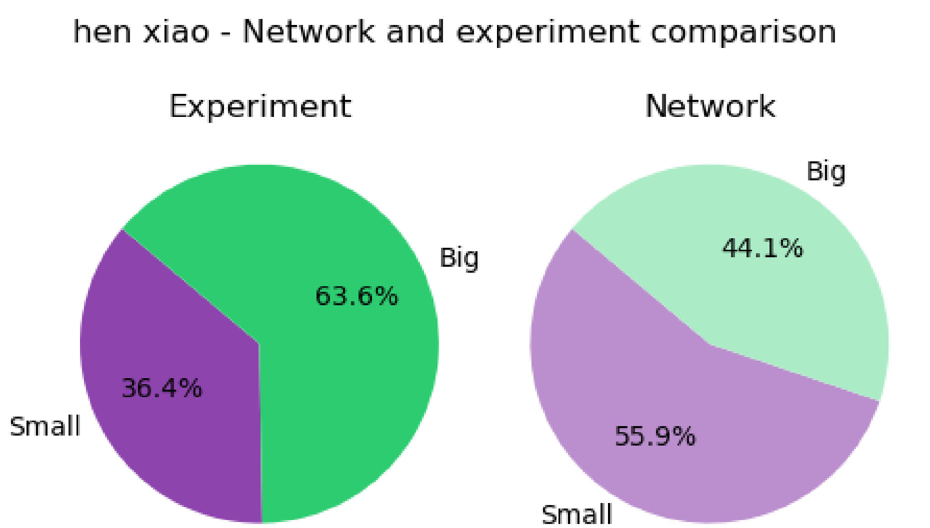

da - Network and experiment comparison
Experiment
Network
Big
59.1%
40.9%
Small
Big
68.5%
31.5%
Small

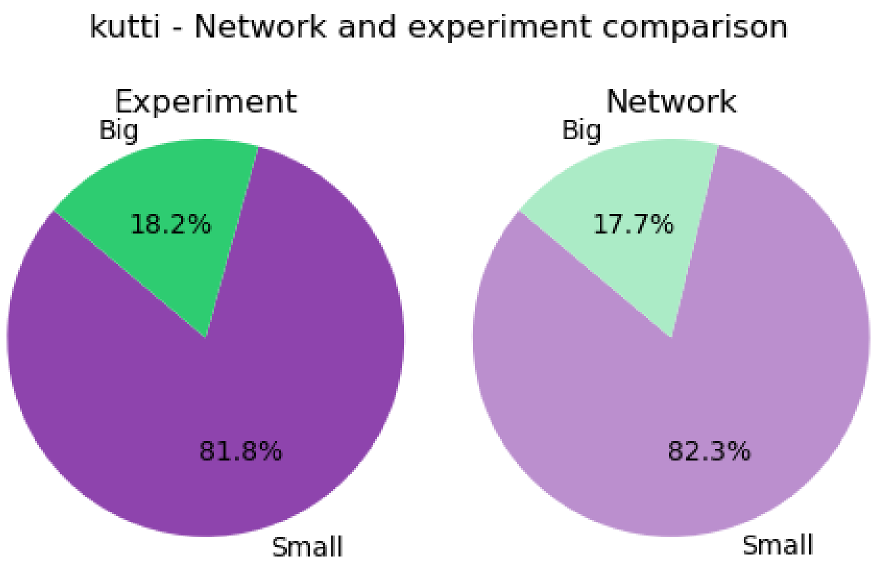

chiru - Network and experiment comparison
Experiment
Network
Big
18.2%
81.8%
Small
Big
46.6%
53.4%
Small

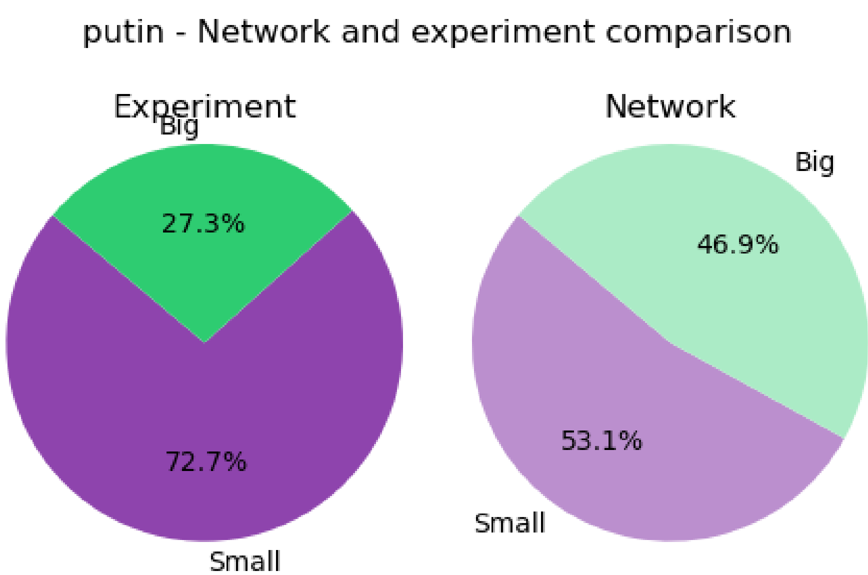

mic - Network and experiment comparison
Experiment
Big
36.4%
63.6%
Small
Network
Small
17.4%
82.6%
Big


kort - Network and experiment comparison
Experiment
Big
31.8%
68.2%
Small
Network
Big
61.4%
38.6%
Small


vigan - Network and experiment comparison
Experiment
Network
Big
63.6%
36.4%
Small
Big
67.6%
32.4%
Small


hen xiao - Network and experiment comparison
Experiment
Network
Big
63.6%
36.4%
Small
Big
44.1%
55.9%
Small





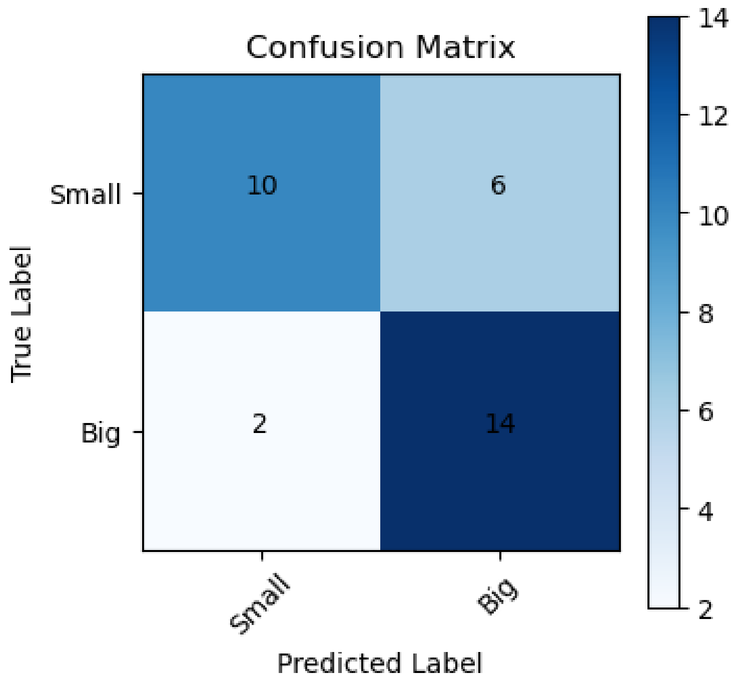
In certain cases the two have a good matching but this verifies not enough often to state a potential correlation.

Finally, let's see the score of the network if it predicts the most probable class. The accuracy score is exactly the same of the "vowel subject" and it performs worse than only one human subject. This may lead us to believe that the network has been able to **capture some significant dependencies to determine the meaning of the word**, even in a language entirely different from that seen during training.

```
[47]: import numpy as np

rnn_subject = Subject(id_="rnn")

for word in words_experiment:

    network_probs = inference.predict(word=word.word)
    pred = np.argmax(network_probs)

    answer = Meaning.SMALL if pred == 0 else Meaning.BIG

    rnn_subject.add_answer(word_id=word.id_, answer=answer)

rnn_experiment = Experiment(subject=rnn_subject, words=words_experiment)
```

Let's see the confusion matrix

```
[48]: rnn_experiment.plot_confusion_matrix()
```




```
[49]: print("Accuracies")
print(f"- experiment: {experiments.mean_score}")
print(f"- network:    {rnn_experiment.score}")
```

```
Accuracies
- experiment: 0.6463068181818182
```

`- network:    0.75`

2.2.13 4. Conclusion

In conclusion, both the experiment and the neural network are toy examples that cannot be considered as significant for drawing conclusions. A more formal experimental setup, network modeling, and training dataset would be required. However, the results of this small experiment lead to the speculation that there may be underlying meaning within the arbitrariness of words.