The size of the Codenames word pool

Jimmy Jin

June 14, 2020

My coworker Christian posted an interesting problem in the company Slack the other day:

> Nerd snipe after playing some Code Names. Imagine you're playing online and after 3 games you've seen a particular word appear in each game. Estimate the total number of words the cards are chosen from. (Each game independently choose 25 distinct words from a fixed set of size N. What approximately is N?)

It's understood that this means there was **exactly** one word which appeared in all three boards, no more. Also, it's allowable for a word to appear in two of the boards but not all three.

How do we estimate this? It turns out that this is actually straightforward using maximum likelihood.

Getting the likelihood

Let G_1, G_2, G_3 be the set of words on the three codenames boards. G_1 and G_2 don't matter. We only require that G_3 contain exactly one word from $G_1 \cap G_2$. Call this event E.

What's the probability of E? It's easier if we break it up. Let F_k be the event that $|G_1 \cap G_2| = k$. Then by the law of total probability,

$$P(E) = \sum_{k=0}^{25} P(E \,|\, F_k) \cdot P(F_k)$$

Computing the probability of each component is straightforward by counting:

- $P(F_k) = \binom{25}{k}\binom{n-25}{25-k}/\binom{n}{25}$
- $P(E \,|\, F_k) = \binom{k}{1}\binom{n-k}{24}/\binom{n}{25}$

In the expression for $P(F_k)$, note that there are $\binom{25}{k}$ ways to choose exactly the k cards in the intersection of G_1 and G_2, and then there are $\binom{n-25}{25-k}$ ways to choose the remaining $25 - k$ cards which are not in the intersection.

Similarly, for $P(E \,|\, F_k)$ there are $\binom{k}{1} = k$ ways to choose exactly one word from $G_1 \cap G_2$ because we have conditioned on F_k, and there are $\binom{n-k}{24}$ ways to choose the remaining 24 words. This can be extended to exactly 2, 3, ... words in the intersection in the obvious way.

Thus the final likelihood is

$$P(E) = \sum_{k=0}^{25} \frac{\binom{k}{1}\binom{n-k}{24}}{\binom{n}{25}} \cdot \frac{\binom{25}{k}\binom{n-25}{25-k}}{\binom{n}{25}}$$

The MLE

I'm not sure if the above expression can be simplified, but I definitely know that I'm too lazy to simplify it. So it wrote a script to just find the argmax:

```
from scipy.special import comb

def intersection_of_k(n, k):
    return comb(25, k) * comb(n-25, 25-k) / comb(n, 25)

def draw_exactly_one_from_intersection(n, k):
    return comb(k, 1) * comb(n-k, 25-1) / comb(n, 25)

def total_prob(n):
    terms = [intersection_of_k(n, k) * \
        draw_exactly_one_from_intersection(n, k) \
        for k in range(1, 25+1)]
    return np.array(terms).sum()

for n in range(100, 200):
    print(n, total_prob(n))
```

This gives an argmax of $n = 125$.

Method of moments

My friend John also pointed out that the MoM estimator for this is also $n = 125$, obtained by solving the equation

$$n \cdot (25/n)^3 = 1$$

Since the left hand side is the expression for the expected number of times a single word will appear in all three boards. Cool.

What else?

It's natural to extend the problem in two ways:

- What's the word pool size if I observe $1 < k < 25$ words appears in all three boards?
- What if I play the game $N > 3$ times and observe a word in exactly all N games?

I think the above approach generalizes to these extensions nicely. The first one is just a simple modification of $P(E \mid F_k)$ discussed above. The second is a little more tricky, but essentially follows the same recipe. In fact, the approach feels like it could be extended in a recursion for that latter case.

If you figure out the answer to these, let me know! Would be curious to visualize the results.