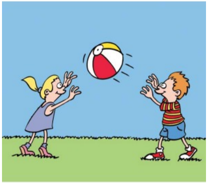
Dissertation update and stats questions

Levi King
Indiana University

October 2020

Recap: Data Collection

I collected native speaker (NS; n=50) and non-native speaker (NNS; n=70) responses to a picture description task (PDT).

10 intransitive items	10 transitive items	10 ditransitive items
		
What is the girl doing?	What is the boy doing?	What is the boy doing?

Recap: Feature annotation

What is the boy doing?	C	A	G	I	V
He is eating food.	0	1	1	1	1
he eating.	0	1	0	1	1
The child is about to eat pizza.	1	0	1	1	1
He may get fat eating pizza.	0	0	1	1	0

Table: Annotated for five features: Core event (C), Answerhood (A), Grammaticality (G), Interpretability (I) and Verifiability (V).

Recap: Feature annotation weights

I used a preference test to establish feature weights. In this toy example, weights are based on 3 pairs. The net score for the preferred responses for each feature is divided by the sum of all 5 net scores (sum=6; e.g., *C* weight: 2/6=.333). The real weights* are based on 1200 pairs across all items.

What is the boy doing?	Pref?	C	A	G	I	V
He is eating food.	**yes**	**0**	**1**	**1**	**1**	**1**
He may get fat eating.	no	0	0	1	1	0
He is hungry.	no	0	0	1	0	1
the boy is eating pizza	**yes**	**1**	**1**	**1**	**1**	**1**
The child is about to eat pizza.	**yes**	**1**	**0**	**1**	**1**	**1**
he eating.	no	0	1	0	1	1
Totals preferred responses		2	2	3	3	3
Totals dispreferred responses		0	1	2	2	2
Net preferred (pref - dispref)		2	1	1	1	1
Feature weight		.333	.167	.167	.167	.167
*Real feature weight		.365	.093	.055	.224	.263

Recap: Gold Standard

I applied the feature weights to the annotations to establish a gold standard (GS) score for each NNS response (n=70) for each PDT item. I ranked by GS score to get a GS ranking. (I use the real weights in this example.)

Participant	*What is the boy doing?*	C	A	G	I	V	GS score	GS rank
p1	The boy is eating.	0	1	1	1	1	0.635	4
p2	A baby is eating pizza	0	0	1	1	0	0.279	5
p3	The boy enjoys his pizza.	1	0	1	1	1	0.907	2
p4	the boy is eating pizza	1	1	1	1	1	1.0	1
p5	The kid is eats pizza	1	0	0	1	1	0.852	3

Recap: Auto scoring

I have a system for automatically scoring the NNS responses. (The details aren't really important here, but ...)

For each item, the process is like this:
For the collection of NS responses (n=50 per PDT item):
1) dependency parse;
2) get tf-idf score for each unique dependency (*Compare against a large balanced corpus; common dependencies get low scores, rare dependencies get higher scores*).

For each NNS response, repeat *1* and *2*, then compare NS vs NNS (dependency scores vectors) – use cosine. This is the NNS response score.

By selecting different parameters in this approach, I arrive at 12 different system configurations. Each configuration scores and ranks all NNS responses (n=70).

Recap: Configurations

Rather than the full set of 12 configurations, let's consider this simplified set of 2 parameters x 2 settings = 4 configurations.

Parameters:

- **Dependency format**:
  - **labeled**: e.g., nsubj(eat,boy); nobj(eat,pizza)
  - **unlabeled**: e.g., ⟨null⟩(eat,boy); ⟨null⟩(eat,pizza)
- **NS response model**: Each NS participant gave *two* responses per PDT item
  - **first**: Model contains only the first response from NS (n=50)
  - **mixed**: Model is half first reponses (n=25) and half second responses (n=25)

dep\model	first	1st & mixed
labeled	lab_first	lab_mixed
unlabeled	unlab_first	unlab_mixed

Table: Four system configurations for scoring NNS responses.

Recap: Gold Standard

I run the NNS responses through my system using the four different configurations. This yields a score and ranking for each response.

P	C	A	G	I	V	GS s	GS r	lf s	lf r	uf r	uf r	lm s	lm r	um r	um r
p1	0	1	1	1	1	0.63	4	.53	4	.11	5	0.29	4	.39	3
p2	0	0	1	1	0	0.27	5	.13	5	.15	4	0.15	5	.53	5
p3	1	0	1	1	1	0.90	2	.91	1	.68	1	0.33	3	.55	1
p4	1	1	1	1	1	1.0	1	.80	2	.41	2	0.70	1	.24	2
p5	1	0	0	1	1	0.85	3	.77	3	.20	3	0.63	2	.22	4

Table: Response scores (*s*) and ranks (*r*) for: gold standard (*GS*); four configurations: labeled_first (*lf*), unlabeled_first (*uf*), labeled_mixed (*lm*), unlabeled_mixed (*um*).

Stats questions

That brings us to where I'm stuck...

- I have three basic categories of PDT item – **intransitive, transitive, ditransitive**. (There is some variation, e.g., *She is riding a bike* vs. *She is bicycling*, but these categories are roughly true.)
- Ideally, I'd like to **find trends** that allow me to **optimize my configuration** for each item category.
- What I've tried:
- I used the GS ranking and the configuration rankings to calculate a single **Spearman** correlation for each configuration, for each item.
- 12 configurations x 30 items = 360 Spearman scores.
- I used these scores to generate hierarchical clusters of items. I did this in nearly every conceivable way; I used: *all* items; *individual* items; I averaged Spearman scores for a given parameter setting, e.g., to compare labeled and unlabeled, I averaged *labeled_first* + *labeled_mixed*, then averaged *unlabeled_first* + *unlabeled_mixed*, then clustered items based on these two sets of values.
- I hoped to find intransitive items clustered together, transitive items clustered together, etc. Any such trends appear very weak, however.

Stats questions

- I need guidance on to how to approach this in a sound way.
- I've begun experimenting with **T-test** and **Wilcox** test. In this case, the idea is to analyze individual features. For example, for a given item and for a given configuration, group all responses where *Core event* is annotated "1", then group all the "0" responses. Then run a **paired sample T-test** using the system score for those groups to see if there are significant differences between them. If I do this for all items, I can look for differences between the intransitive, transitive, ditransitive items across all configurations.
- An important note here – the **feature annotations are heavily skewed**. For a handful of the (30 items x 5 features =) 150 cases, a feature is "1" for *all* responses.
- I'm also considering this approach but using **average precision** instead of T-test. In this case, I'd be looking for configurations that maximize the separation of "0" and "1" responses.

References