An Investigation of Improving Data Classification with Natural Language

Predicting Baseball Plate Appearance Outcomes

August Milliken
Department of Linguistics, Department of Computer Science
University of Colorado, Boulder

Goals and Motivation

In this project we attempt to **predict baseball plate appearance outcomes over 11 classe**s from just numerical data.

To accomplish our goal, we:

- **Explore a novel approach**, of exploiting the power of **NLP representations to boost the learning** of an encoder
- **Preform an ablation** to attempt and confirm the intuition of the ability of the natural language
- Propose a **potentially more cost-effective** alternative to simply using a massive model

This was born of the belief that natural language's complex representation might help guide a deep classifier

This approach could potentially be applied to similar tasks

Taking Advantage of NLP

- Word and **sentence embeddings are rich in meaning** often in a large dimensional space
- **Pretrained embeddings provide more meaning** to words and sentences
- **Embeddings' semantic similar**, or dissimilarity, should be replicated by their **distance in the representational space**
- These embeddings might be **used as "guardrails"** for how our encoder module learns during training

Data and Methodology

- The Statcast data is pulled using PyBaseball covering every game for the last **10 years**
- Data is at a **pitch-by-pitch** level with 100 data points per pitch and distilled down to **47 usable data points**
- The descriptive sentences were processed with **MiniLM-L6-v2**

- Input features: game id, game date, home team, away team, pitch name, pitcher handedness, release x y z, fielding alignment, etc

- Classes: Out, single, double, homerun, multi-out, field play, triple, strikeout, awarded first, walk, other

There were many architectures tested the final versions were:

- **Encoder:**

Input: Game state embedding
Output: False sentence embedding
Design: Transformer encoder treating each stat as a separate token with skip connections and layer normalizations

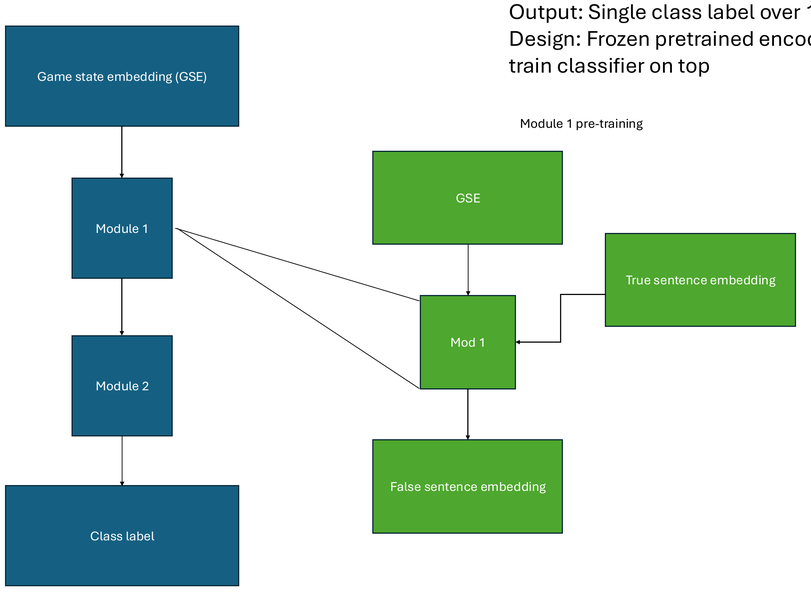
- **Classifier:**

Input: Sentence embedding (with or without the game state concatenated) or game state
Output: Single class label over 11 labels
Design: Feedforward classifier with layer normalizations

- **Full model:**

Input: Game state embedding
Output: Single class label over 11 labels
Design: Frozen pretrained encoder model and train classifier on top

Full model design



Side of the plate batter is standing on	X-axis Release Position	NL Description	Label
1	-2.44	Zack Gelof pops out to shortstop Masyn Winn.	Out
1	0.88	Dylan Moore strikes out swinging.	Strike out
0	-1.24	Jorge Polanco walks.	Walk

Example data with only 2 of the features

Results

	Macro precision	Weighted precision	Cosine similarity
Encoder only	N/A	N/A	71.2%
Sentence classifier	99.9%	99.9%	N/A
Game state classifier	40.3%	56.6%	N/A
Stacked model PF	43.3%	61.2%	N/A
Stacked model P	40.1%	56.6%	N/A
Stacked model	40.3%	56.6%	N/A

The stacked model is the best resulting version where it uses a pretrained encoder that has its weights frozen, where the final architecture is outlined in the model figure; P meaning pretrained encoder, F meaning Frozen encoder weights

Conclusions and Discussion

- This method does offer a boost in classification but it **does not seem to be as productive as we would expect**
- The **struggle may lay in the encoder** as we know that more meaningful representations are classified well
- It may also benefit from more hyper-tuning
- Having **player representations** could be massively helpful (potentially to the simple classifier as well)
- **PCA** on the input to assure we are optimizing training

References