Title: Something snappy about the SLAM competition

Alexander S. Rich **Pamela Osborn Popp** **David J. Halpern**
Anselm Rothe **Todd M. Gureckis**
Department of Psychology, New York University
{asr443,pamop,david.halpern,anselm,todd.gureckis}@nyu.edu

Abstract

1 Introduction

Blah Blah (Settles et al., 2018)

2 Task Approach

We approached the task as a binary classification problem over instances (i.e., single words within an exercise). Our solution can be divided into two components—constructing a set of features that is highly informative about whether the user will answer an instance correctly, and designing a model that can achieve high performance using this feature set.

2.1 Feature Engineering

We used a variety of features, including features directly present in the training data, features constructed using the training data, and features that use information external to the training data. Except where otherwise specified, categorical variables were one-hot encoded.

2.1.1 Exercise features

We encoded the exercise client, session, format, and time (i.e., number of seconds to complete the exercise), as well as the exercise number and number of days since start of usage.

2.1.2 Word features

Using spaCy[1], we lemmatized each word to produce a root word. Both the root word token and the original token were used as categorical features. Due to their high cardinality, these features were not one-hot encoded but were preserved in single columns and handled in this form by the model (as described below).

Along with the tokens themselves we encoded instance word's part of speech, morphological features, and dependency edge label. (We noticed that some words in the original dataset were paired with the wrong morphological features, particularly near where punctuation had been removed from the sentence. To fix, this, we re-processed the data using Google SyntaxNet[2].)

We also encoded word length and several word characteristics gleaned from external data sources. PAM AND DAVID FILL IN HERE WITH FREQ, LEVENSHTEIN, AND AOA.

2.1.3 User features

Just as we did for word tokens, we encoded the user ID as a single-column, high-cardinality feature. We also calculated several other user-level features. TODD AND ANSELM HERE

2.1.4 Positional features

To account for the effects of surrounding words on the difficulty of an instance, we created several features related to the instance word's context in the exercise. These included the token of the previous word, the next word, and the instance word's root in the parse tree, all stored in single columns as with the instance token itself. We also included the part of speech of each of these context words as additional features. When there was no previous word, next word, or parse root word, a special `None` token or `None` part of speech was used.

2.1.5 Temporal features

A user's probability of succeeding on an instance is likely related to their prior experience with that instance. To capture this, we calculated several features related to past experience. We encoded the number of times the current exercise's exact sentence had been seen before by the user. is

[1] https://spacy.io/

[2] https://github.com/ljm625/syntaxnet-rest-api

this right Todd? We also encoded a set of features recording past experience with the particular instance word. These features were encoded separately for the instance token and for the instance root word created by lemmatization.

For each token (and root) we tracked user performance through four weighted error averages. At the user's first encounter of the token, each error term E starts at zero. After an encounter with an instance of the token with label L, it is updated according to the equation

$$E \leftarrow E + \alpha(L - E)$$

where α determines the speed of error updating. The four feature weighted error terms use $\alpha = \{.3, .1, .03, .01\}$, allowing both short-run and long-run changes in a user's error rate with a token to be tracked. Note that in cases where a token appears multiple times in an exercise, a single update of the error features is conducted using the mean of the token labels. Along with the error tracking features, for each token we calculated the number of {total, labeled, unlabeled} encounters, time since last encounter and labeled encounter, and whether the instance is the first encounter with the token.

In the training data, all instances are labeled as correct or incorrect, so the label for the previous encounter is always available. In the test data, labels are unavailable, so predictions must be made using a mix of labeled and unlabeled past encounters. To generate training-set features that are comparable to test-set features, we selectively ignored some labels when encoding temporal features on the training set. Specifically, for each user we first calculated the number of exercises n in the true test set. Then, when encoding the features for each instance, we selected a random integer r in the range $[1, n]$, and ignored labels in the prior r exercises. That is, we encode features for the current instance as though other instances in those prior exercises were unlabeled, and ignore updates to the error averages from those exercises. The result of this process is that each instance in the training set is encoded as though it were between one and n exercises into the test set.

2.2 Modeling

3 Feature Removal Experiments

References

B. Settles, C. Brust, E. Gustafson, M. Hagiwara, and N. Madnani. 2018. Second language acquisition modeling. In *Proceedings of the NAACL-HLT Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*. ACL.