Faster Feature Engineering by Approximate Evaluation

Michael R. Anderson  
University of Michigan  
mrander@umich.edu

Michael Cafarella  
University of Michigan  
michjc@umich.edu

ABSTRACT
The application of machine learning to large datasets has become a vital component of many important and sophisticated software systems built today. Such trained systems are often based on supervised learning tasks that require features, or extracted signals that distill complicated raw data objects into a small number of salient values. A trained system’s success depends on the quality of its features.

Unfortunately, feature engineering—writing code that turns raw data objects into feature vectors suitable for a machine learning algorithm—is tedious and time-consuming. Because “big data” inputs are so diverse, feature engineering is a trial-and-error process with many small, iterative code changes. Because the inputs are so large, each code change can involve time-consuming data processing (over each page in a Web crawl, for example). We introduce Zombie, a data-centric system that accelerates feature engineering through intelligent input selection, optimizing the “inner loop” of the feature engineering process. Our system yields feature evaluation speedups of up to 8x in some cases and reduces engineer wait times from 8 to 5 hours in others.

1. INTRODUCTION
The ultimate success of a trained system depends on the ability of features to accurately represent the task-specific variations within the raw data. Commodity features, such as token counts, are often the first attempt at developing a feature set. However, these features are by necessity general: they cannot take advantage of nuances specific to the data or task. Using commodity features as a starting point, a feature engineer can develop additional features tailored to the particular dataset by applying domain expertise [2, 6].

Feature engineering can be a highly iterative, trial-and-error process, because, unfortunately, good features are hard to devise. First, with a large, diverse set of inputs (e.g., a Web crawl), the engineer never quite knows the “input specification” and must repeatedly fix bugs as unexpected raw inputs are discovered. Second, predicting the usefulness of a proposed feature is difficult; a programmer may implement a feature only to throw it away after finding it ineffective. Feature engineers, then, often must make and evaluate many small iterative changes to their code. But evaluating each feature code change can take a great deal of time, since it entails executing the feature code over a very large set of raw inputs.

Deep learning methods have great promise for producing high-quality models without traditional explicit feature engineering [7]. However, we believe there will always be a strong role for human-provided domain knowledge.

In this development model, the engineer can face significant downtime in Step 2, waiting for the feature code to be applied to a large dataset. Even with a distributed processor like MapReduce or Spark, executing complex feature code over millions of data items takes considerable time. Our research focuses on the efficient development and execution of feature functions over very large raw datasets (Steps 1 and 2).

We have implemented Zombie, a system that saves considerable time in the above development cycle by reducing the time cost of generating the features for a suitable training set (Step 2) [3, 4]. Like giving a developer a faster compiler, these savings allow the feature engineer to iterate on feature code quickly; because of this improved engineer productivity, tasks can be completed faster or with a higher quality outcome than under standard development processes.
we can limit the processing of the data to these items, a
the feature generation runtime that we are trying to reduce.
methods examine the features of potential items, entailing
active learning [8]. Unfortunately, standard active learning
learning task and to the engineer’s feature functions. If
query answer by only generating features for a subset of the
features, and then stop processing the raw data early once
system as the training set is populated with newly processed
portion is to continuously re-train and evaluate the learning
is to continuously re-train and evaluate the learning
function already in use. Unfortunately, using just a subset of a large training corpus is a common
raw data and training the learning system on this reduced
query by only generating features for a subset of the
features used for a machine learning task are essential
To avoid this limitation, our system pre-processes the
raw data, creating a large number of index groups, each
containing raw data items similar to one another. Akin to
the offline sample creation of approximate query processing
systems like BlinkDB [1], ZOMBIE uses raw data items in the
highest utility index groups to generate features to answer
the evaluation query. Unlike approximate query processors,
we cannot know which index groups will be most useful to a
particular feature function’s evaluation query. A raw data
item may produce useful features under some functions, but
be irrelevant under others. Thus, we must learn the utility of
the index groups online for each evaluation query.
Online learning has a tradeoff between exploration and
exploitation: we want to select items from the best input
groups, but we must also process items from unknown—and
potentially suboptimal—index groups to learn which are best.
ZOMBIE uses a multi-armed bandit algorithm to manage this
tradeoff: a raw data item is randomly drawn from an index
group (selected by the UCB bandit algorithm [5]), processed
to generate features for the training set, and used to re-train
the learning system. The index group’s utility is updated
based on the incremental change in the model’s quality.
This method of raw input selection is shown by the ZOMBIE
scan learning curve in Figure 2. The point labeled ZOMBIE
shows how effective this method is at reducing the feature
generation runtime over a Bulk scan method, even with early
stopping. Our experiments (described in our full paper [3])
have shown up to an 8X speedup over the Bulk scan with
early stopping in some cases, as well as reducing engineer
wait time from 8 to 5 hours in others, on a series of text
classification tasks. ZOMBIE’s operation is orthogonal to
statistical improvements in learning algorithms, and so could
be easily combined with advances in that area.
4. CONCLUSION
The features used for a machine learning task are essential
for a successful trained system, and engineering great fea-
tures is a difficult task. Our research aims to remove some of
the tedium and unproductive thumb twiddling inherent in
the current practice of feature engineering by providing tools
to find the right features for the right data. By intelligently
selecting raw data to process for feature generation, ZOMBIE
allows the feature engineer to quickly evaluate feature effec-
tiveness for a given learning task, improving productivity
and the potential for building a high-quality trained system.
REFERENCES
[1] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and
I. Stoica. BlinkDB: Queries with bounded errors and bounded
response times on very large data. In EuroSys, 2013.
An integrated development environment for faster feature