Multimodal AutoML on Tables with Text Fields

Xingjian Shi* xjshi@amazon.com
Jonas Mueller* jonasmue@amazon.com
Nick Erickson neerick@amazon.com
Mu Li mli@amazon.com
Alexander J. Smola alex@smola.org
Amazon Web Services

Abstract

We consider the design of automated supervised learning systems for data tables that not only contain numeric/categorical columns, but text fields as well. Here we assemble 15 multimodal data tables that each contain some text fields and stem from a real business application. Over this benchmark, we evaluate numerous multimodal AutoML strategies, including standard two-stage approaches where NLP is used to featurize the text such that AutoML for tabular data can then be applied. We identify practically superior strategies based on multimodal adaptations of Transformer networks and stack ensembling of these networks with classical tabular models. Compared with human data science teams, the best fully automated methodology discovered through our benchmark manages to rank 1st place when fit to the raw text/tabular data in two MachineHack prediction competitions and 2nd place (out of 2380 teams) in Kaggle’s Mercari Price Suggestion Challenge.

1 Introduction

Despite recent data proliferation, the practical value of machine learning (ML) remains hampered by an inability to quickly translate raw data into accurate predictions. Automatic Machine Learning (AutoML) aims to address this via pipelines that can ingest raw data, train models, and output accurate predictions, all without human intervention [35]. Given their immense potential, many AutoML systems exist for data structured in tables, which are ubiquitous across science/industry [23, 30, 57].

Many data tables contain not only numeric and categorical fields (together referred to as tabular here), but also fields with free-form text. For example, Table 1 depicts actual data from the website Kickstarter. These contain multiple text fields such as the title and description of each funding proposal, numerical fields like the goal amount of funding and when the proposal was created, as well as categorical fields like the funding currency or country. Despite their potential commercial value, there are currently few (automated) solutions for machine learning with such multimodal data.

Applying existing AutoML tools to such data thus requires either manually featurizing text fields into tabular format [4, 28], or ignoring the text. Alternatively, one can use existing natural language processing (NLP) tools to model primarily just the text [10, 26, 27, 34, 51].

This paper considers design choices for automated supervised learning with multimodal datasets that jointly contain text, numeric, and categorical features. Even though text commonly appears along with numeric/categorical fields in enterprise data tables, how to automatically analyze such multimodal data has not been well studied in the literature. This stems from a lack of published benchmarks, as well as existing beliefs that basic featurization of the text [14, 28] should suffice for

*Equal contribution.

Now available to easily run on your own data through: https://github.com/awslabs/autogluon

tabular models to exhibit strong performance. Here we introduce a new benchmark of 15 multimodal text/tabular datasets from real business applications, and provide the first comprehensive evaluation of generic strategies for supervised learning with such data. Among other discoveries, our benchmark reveals that the conventional modeling strategy of using deep (contextual) embeddings to featurize text for tabular models is far from optimal. We hope the public benchmark and open-source tooling introduced here will spur further research in this important practical direction.

2 Related Work

Often, ML courses teach data as vectors in $\mathbb{R}^d$, which is not the case in many practical applications. Substantial research has been needed to properly handle categorical features in a unified manner that generalizes across datasets without sacrificing accuracy [25, 42, 48]. Beyond categorical variables, extensive effort has been devoted to adapt ML models for text, a highly-structured form of data with unique properties like variable-length and transferable information shared across corpora [14]. Despite an urgent need, tools for fully automated learning with text data currently remain scarce (e.g. this dearth forced Blohm et al. [4] to turn to tabular AutoML tools for automated text prediction). Instead NLP applications today primarily require experts who have unanimously come to favor Transformer networks as their model of choice for text [13, 49, 51]. However existing methods to input numeric/categorical features into Transformer models remain quite rudimentary [51] and fail to outperform the best tree algorithms for tabular prediction [33]. Note that while seemingly similar, recent work on Transformers for understanding structured tables [12, 66] addresses different tasks than the multimodal supervised learning studied in this paper.

The use of tabular models together with Transformer-like text architectures has also only received limited attention [39, 61], and it remains unclear how to optimally leverage their complementary strengths for multimodal data (due to lack of benchmarks). In contrast, a number of entirely-neural architectures have been proposed for multimodal settings [36, 52, 53, 64]. However the vast majority of these are for \{image, text\} data [2, 50, 54, 55], but the gap between neural networks and alternative models is far greater for images than for tabular data [33]. Beyond the open-source packages studied in this paper, the few other existing tools that aim to automate multimodal text/tabular ML are all commercial software whose source code, allowed scientific usage (benchmarking), and implemented algorithmic strategies remain opaque [24, 29].

Large-scale public benchmarks that are sufficiently diverse/representative have been responsible for significant progress in tabular AutoML [15, 16, 23, 68] and NLP [21, 40, 47, 63]. However we are not aware of any analogous benchmarks for evaluating multimodal text/tabular ML. There do exist a few miscellaneous text/tabular datasets scattered throughout popular ML data repositories [1, 59], but these are mostly small academic datasets that are not representative of modern text/tabular ML applications with significant practical value. In contrast, multiple prediction competitions each involving a single real-world text/tabular dataset have been held, but winning solutions have heavily relied on dataset/domain-specific tricks [45]. Here we aggregate multimodal datasets from such competitions and other industry sources into a single benchmark that aims to reveal unifying principles for powerful generic modeling of this form of data.

<table>
<thead>
<tr>
<th>name</th>
<th>desc</th>
<th>goal</th>
<th>country</th>
<th>currency</th>
<th>created_at</th>
<th>final_status</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Secret Order - The Game that gives back G...</td>
<td>Can you trust your friends? Solve the puzzle? ...</td>
<td>5000.0</td>
<td>GB</td>
<td>GBP</td>
<td>1424101105</td>
<td>0</td>
</tr>
<tr>
<td>Booker Family Foods. Home made, the way food s...</td>
<td>Community based, home-made-foods producer, to ...</td>
<td>2500.0</td>
<td>US</td>
<td>USD</td>
<td>1404617242</td>
<td>0</td>
</tr>
<tr>
<td>J.A.E.S.A - Next Generation Artificial Intell...</td>
<td>A true next generation AI with the ability to ...</td>
<td>30000.0</td>
<td>CA</td>
<td>CAD</td>
<td>1399078600</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1: Example of data in our multimodal benchmark with text (name, desc), numeric (goal, created_at), and categorical (country, currency) columns. From these features, we want to predict if a Kickstarter project will reach its funding goal or not (final_status).
3 Benchmarking Multimodal Text/Tabular AutoML

We aim to design practical systems for real-world data tables that often contain text. The empirical performance of our design decisions is thus what ultimately matters. Representative benchmarks comprised of many diverse datasets are critical for proper evaluation of AutoML, whose aim is to reliably produce reasonable accuracy on arbitrary datasets without manual user-tweaking. Thus we introduce the first public benchmark for evaluating multimodal text/tabular ML, which is comprised of 15 tabular datasets, each containing at least one text field in addition to numeric/categorical columns. Our new benchmark is publicly available, as is the code to reproduce all results presented in this work (and also to recreate our modified benchmark datasets from the original data sources).

Our benchmark strives to represent the types of ML tasks that commonly arise in industry today. In creating the benchmark, we aimed to include a mix of classification vs. regression tasks and datasets from real applications (as opposed to toy academic settings) that contain a rich mix of text, numeric, and categorical columns. Table 2 shows it is comprised of datasets that are quite diverse in terms of: sample-size, problem types, number of features, and type of features. To reflect real-world ML issues, we processed the data minimally (beyond ensuring the features/labels correspond to meaningful prediction tasks without duplicate examples) and thus there are arbitrarily-formatted strings and missing values all throughout. Subsequent accuracy results from Table 3 indicate the 15 underlying prediction problems also vary greatly in terms of both difficulty and how the predictive signal is divided between text/tabular modalities. We caution our benchmark only contains text in the English language and primarily from commercial domains. Thus its conclusions will only hold for particular types of applications. Nonetheless, systems that can perform well across the diverse set of 15 benchmark datasets are likely to provide real-world value for an important class of applications.

Each dataset in our benchmark is provided with a prespecified training/test split (usually 20% of the original data reserved for test set). Methods are not allowed to access the test set during training, and for validation (model-selection, hyperparameter-tuning, etc.) instead must themselves hold-out some data from the provided training data. As the choice of training/validation split is a key design decision in AutoML, we leave this flexible for different systems to choose as they see fit. To facilitate comparison between the novel AutoML strategies presented in this paper, we always used the same AutoGluon-provided training/validation split, which is stratified based on labels in classification tasks. Our use of other AutoML frameworks beyond AutoGluon (e.g. H2O) allows each framework to choose their own data splitting scheme.

4 Multimodal Modeling Strategies

We first outline the many possibilities that must be considered in AutoML for multimodal data tables with text. Key design choices include what models to use (and for which features), and how to optimally combine different models within an overall ML pipeline. Our study aims to cover the major different modeling paradigms used by practitioners today, including: NLP models to featurize text for tabular models [4, 14, 28], ensembling of independently-trained text and tabular models [45], or end-to-end learning with neural networks that operate across all modalities [36, 51, 52].

<table>
<thead>
<tr>
<th>Dataset ID</th>
<th>#Train</th>
<th>#Test</th>
<th>#Cat.</th>
<th>#Num.</th>
<th>#Text</th>
<th>Task</th>
<th>Metric</th>
<th>Prediction Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>prod</td>
<td>5,091</td>
<td>1,273</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>multiclass accuracy</td>
<td>sentiment associated with product review</td>
<td></td>
</tr>
<tr>
<td>airbnb</td>
<td>18,316</td>
<td>4,579</td>
<td>37</td>
<td>24</td>
<td>28</td>
<td>multiclass accuracy</td>
<td>price of Airbnb listing</td>
<td></td>
</tr>
<tr>
<td>channel</td>
<td>20,284</td>
<td>5,071</td>
<td>1</td>
<td>15</td>
<td>1</td>
<td>multiclass accuracy</td>
<td>news category to which article belongs</td>
<td></td>
</tr>
<tr>
<td>wine</td>
<td>84,123</td>
<td>21,031</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>multiclass accuracy</td>
<td>which variety of wine</td>
<td></td>
</tr>
<tr>
<td>imdb</td>
<td>800</td>
<td>200</td>
<td>0</td>
<td>7</td>
<td>4</td>
<td>binary roc-auc</td>
<td>whether film is a drama</td>
<td></td>
</tr>
<tr>
<td>jigsaw</td>
<td>100,000</td>
<td>25,000</td>
<td>2</td>
<td>27</td>
<td>1</td>
<td>binary roc-auc</td>
<td>whether social media comments are toxic</td>
<td></td>
</tr>
<tr>
<td>fake</td>
<td>12,725</td>
<td>3,182</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>binary roc-auc</td>
<td>whether job postings are fake</td>
<td></td>
</tr>
<tr>
<td>kick</td>
<td>86,502</td>
<td>21,626</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>binary roc-auc</td>
<td>whether proposed Kickstarter project will achieve funding goal</td>
<td></td>
</tr>
<tr>
<td>ae</td>
<td>22,662</td>
<td>5,666</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>regression R^2</td>
<td>price of American-Eagle inner-wear items on their website</td>
<td></td>
</tr>
<tr>
<td>qa</td>
<td>4,863</td>
<td>1,216</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>regression R^2</td>
<td>subjective type of answer (in relation to question)</td>
<td></td>
</tr>
<tr>
<td>qaa</td>
<td>4,863</td>
<td>1,216</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>regression R^2</td>
<td>subjective type of question (in relation to answer)</td>
<td></td>
</tr>
<tr>
<td>cloth</td>
<td>18,778</td>
<td>4,699</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>regression R^2</td>
<td>customer review score for clothing item</td>
<td></td>
</tr>
<tr>
<td>mercant</td>
<td>100,000</td>
<td>25,000</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>regression R^2</td>
<td>price of Mercant online marketplace products</td>
<td></td>
</tr>
<tr>
<td>jc</td>
<td>10,860</td>
<td>2,715</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>regression R^2</td>
<td>price of JC Penney products on their website</td>
<td></td>
</tr>
<tr>
<td>pop</td>
<td>24,007</td>
<td>6,002</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>regression R^2</td>
<td>online popularity of news article</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: The 15 multimodal datasets that comprise our benchmark. ‘#Cat.’, ‘#Num.’ and ‘#Text’ count the number of categorical, numeric, and text features in each dataset, and ‘#Train’ (or ‘#Test’) count the training (or test) examples. In PDF, click on each Dataset ID for link to original data source.
Featurizing Text for Tabular Models  In this paper, all tabular (numeric/categorical) modeling is simply done via AutoGluon-Tabular, a highly accurate open-source tool for automated supervised learning on tabular data [15, 67]. AutoGluon achieves strong performance by ensembling a diverse suite of high-quality models for tabular data, including: multiple variants of Boosted Trees [8, 38, 48], Random Forests [5], Extremely Randomized Trees [22], K-Nearest-Neighbors [11], and fully-connected Neural Networks [15]. While neural networks are typically favored for unstructured data like text, decision tree ensembles have proven to be one of the most consistently performant models for tabular data [3, 18, 33]. While deploying home-grown ensembles can be tricky, AutoGluon automatically constructs and deploys its ensembles without any engineering overhead for the user. For real-time applications with latency constraints, AutoGluon provides many options to accelerate ensemble inference via pruning or distillation [18]. Since we have contributed the ensembling techniques of this paper into AutoGluon, our strategies can be utilized with all of the same benefits. To allow tabular models to access information in text fields, the text is typically first mapped to a continuous vector representation which replaces a text column in our data table with multiple numeric columns (one for each vector dimension). One can treat each text column as a document, and each individual text field as a paragraph within the document, such that each text field can be featurized via NLP methods for computing text representations [13, 14, 46, 53].

Transformer Models for Text  Pretrained Transformers have become a cornerstone of modern NLP, where the model is first pretrained in an unsupervised manner on a massive text corpus before being applied to our (smaller) labeled dataset of interest [13, 51]. This allows our modeling to leverage information gleaned from the external text that would otherwise not be available in our limited labeled data. The Transformer also effectively aggregates information from various aspects of a training example, using a self-attention mechanism to contextualize its intermediate representations based on particularly informative features [60]. Since BERT [13] first demonstrated the power of Transformer pretraining via Masked Language Modeling (MLM), superior pretraining techniques have been developed. RoBERTa [44] dynamically generates masks and pretrains on a larger corpus for a longer time, employing the same MLM objective as BERT in which random tokens are masked for the Transformer to guess their original value. ELECTRA [9] is an alternative pretraining technique in which a simple generative model randomly replaces tokens and the Transformer must classify which tokens were replaced. Given multiple text columns, we feed the tokenized text from all columns jointly into our Transformer with special [SEP] delimiter tokens between fields and a [CLS] prefix token appended at the start [13]), as detailed in Appendix A.2. A single embedding vector for all text fields is obtained from the Transformer’s representation at its [CLS] position after feeding the merged input into the network [13]. Similarly, just a single text field can be embedded via the Transformer’s representation at the [CLS] position, after feeding only this field into the network.

Neural Architectures for Multimodal Data  In many multimodal datasets, some of the predictive signal solely resides in text fields, while other predictive information is restricted to tabular feature...
values, or complex interactions between text and tabular values. To enjoy the benefits of end-
to-end learning without sacrificing accuracy, we consider how to adapt Transformer networks to
simultaneously operate on inputs from both modalities. A natural approach in our setting is to enhance
the Transformer such that its attention mechanism can contextualize representations of individual
text tokens based not only on other parts of the text, but also on the values of relevant tabular features
as well. Below we discuss various implementations of this strategy (with details in Appendix A.3).

**All-Text** A simple (yet crude) option is to convert numeric and categorical values to strings and
subsequently treat their columns also as text fields [51]. Through its byte-pair encoding, a pretrained Transformer can handle most categorical strings and may be able to crudely represent numeric values
within a certain range (here we round all numbers to 3 significant digits in their string representation).

**Fuse-Early** Rather than casting them as strings, we can allow our model to adaptively learn token
representations for each numeric and categorical feature via backpropagation (see Figure 1b). We
introduce an extra factorized embedding layer [25, 41] to map categorical values into the same \( \mathbb{R}^d \)
vector representation encoded by the pretrained NLP model for text tokens (with different embedding
layers used for different categorical columns in the table). All numeric features are encoded via
a single-hidden-layer Multi-layer Perceptron (MLP) to obtain a unified \( \mathbb{R}^d \) vector representation.
The resulting \( d \)-dimensional vector representations from each modality are jointly fed into a 6-layer
Transformer encoder whose self-attention operations can model interactions between the embeddings
of text tokens, categorical values, and numeric values. We refer to this strategy as **Fuse-Early** because
only a minimal (yet adaptive) input processing layer is added to convert the tabular features into a
common vector form which can be jointly fed through many shared Transformer layers. Huang et al.
[33] considered a similar strategy for applying Transformers to entirely numeric/categorical data,
even without text components that are a major focus here.

**Fuse-Late** Rather than aggregating information across modalities in early network layers, we can
perform separate neural operations on each data type and only aggregate representations near the
output layer (see Figure 1c). This multi-branch design allows each branch to extract higher-level
representations of the values from each modality, before the network needs to consider how modalities
should be fused. Here we use a multi-tower architecture in which numeric and categorical features are
fed into separate MLPs for each modality. The text features are fed into a (pretrained) Transformer
network. The topmost vector representations of all three networks are pooled into a single vector (via
either: mean/max pooling or concatenation) from which predictions are output via two dense layers.

## 5 Aggregating Text & Tabular Models

Despite their success for modeling text, the application of Transformer architectures to tabular
data remains limited [17, 18, 33]. The use of tabular models together with Transformer-like text
architectures has also received little attention [39, 61]. Note that ‘tabular models’ throughout are
trained on only numeric/categorical features, e.g., various tree ensembles used in AutoGluon-Tabular.

### 5.1 Embedding Text as Tabular Features

In our first class of aggregation methods, a Transformer is used to map the text fields into a vector
representation. Subsequently, the text fields are replaced in the data table by additional columns
corresponding to each dimension of the embedding vector (**Embedding-as-Feature** in Figure 2a). We
consider three ways to featurize text using a Transformer.

**Pre-Embedding** The most straightforward is to use a pretrained Transformer (not fine-tuned on
our labeled data), and subsequently train tabular models over the featurized data table [4].

**Text-Embedding** The Pre-Embedding strategy is not informed about our particular prediction
problem and the domain of the text data. In Text-Embedding, we further fine-tune the pretrained
Transformer to predict our labels from only the text fields. By adapting to the domain of the problem,
Text-Embedding is able to extract more valuable features that can improve the performance of tabular
models. This is particularly true in settings where the target only depends on one out of many text
fields, since the fine-tuning process can produce representations that vary more based on the relevant
field vs. irrelevant text.
Multimodal-Embedding  Text representations may improve when self-attention is informed by context regarding numeric/categorical features. Thus we also consider embedding text via our previous multimodal networks. These models are again fine-tuned using the labeled data and now produce a single vector representation for all columns in the dataset, regardless of their type. Since Transformers are better suited for modeling text than tabular features, we only replace the text fields with the learned vector, all other non-text features are kept and used for subsequent tabular learning.

5.2 Ensembling Text/Tabular Predictions

Utilized by most AutoML frameworks [15, 20, 43], model ensembling is a straightforward technique to boost predictive accuracy. Ensembling is particularly suited for multimodal data, where different models may be trained with different modalities. However, the resulting ensemble may then be unable to exploit nonlinear predictive interactions between features from different modalities. To remedy this, we advocate for the use of our multimodal Transformers that fuse information from text and tabular inputs. Furthermore, we propose stack ensembling with nonlinear aggregation of model predictions that can exploit inter-modality interactions between different base models’ predictions, even when base models do not overlap in modality.

Weighted-Ensemble We first consider straightforward aggregation via a weighted average of the predictions from our Transformer model and various tabular models (like those trained by AutoGluon-Tabular). Here, our Transformer and other models are independently trained using a common training/validation split. Subsequently, we apply ensemble selection, a greedy forward-selection strategy to fit aggregation weights over all models’ predictions on the held-out validation data [7].

Stack-Ensemble Rather than restricting the aggregation to a linear combination, we can use stacking [65]. This trains another ML model to learn the best aggregation strategy. The features upon which the ‘stacker’ model operates are the predictions output by all base models (including our Transformer), concatenated with the original tabular features in the data. Following Erickson et al. [15], we try each type of tabular model in AutoGluon-Tabular as a stacker model. To output predictions, a weighted ensemble is constructed via ensemble selection applied to the tabular stacker models (Figure 2c). We do not consider our larger Transformer model as a stacker since lightweight aggregation models are preferred in practice. Overfitting is a key peril in stacking, and we ensure that stacker models are only trained over held out predictions produced from base models via 5-fold cross-validation (bagging) [15, 58].

6 Experiments

Here we empirically evaluate several multimodal AutoML strategies, with a particular focus on how to best leverage Transformers for text/tabular data. To keep our study tractable, we adopt a sequential decision making process that decomposed our design into three stages: 1) determine the appropriate Transformer backbone and fine-tuning strategy for text data alone, 2) determine the best way for generalizing Transformer to multimodal data among our considered variants, and 3) choose the best method to aggregate text and tabular models. At each subsequent stage of the study, we explore modeling choices that are specific to that stage and simply use the best choice found in the empirical comparisons of the options available in previous stages. Each modeling strategy is run over our benchmark of 15 tabular datasets with text fields, detailed in Appendix 3. We evaluate regression tasks via the coefficient of determination $R^2$, multiclass classification tasks via accuracy, and binary classification tasks via area under the ROC curve (AUC).
We first fine-tune the pretrained Transformer models as our sole predictors, using only the text features in each dataset. This helps identify which model is better at handling the types of text in our multimodal datasets. In addition, we consider two fine-tuning tricks to boost performance: 1) Exponentially decay the learning rate of the network parameters based on their depth $d$ in which $d$ is the layer depth and $\tau$ is the decay factor. 2) Average the weights of the models loaded from the top-3 training checkpoints with the best validation scores [60].

The first section of Table 3 shows that ELECTRA performs better than RoBERTa across the text columns in our benchmark datasets. Our Text-Net used in subsequent experiments is thus ELECTRA fine-tuned with both exponential decay and checkpoint-averaging.

### Best Multimodal Network
Next, we explore the best way to extend the Text-Net model to operate across numeric/categorical inputs in addition to text fields. Three multimodal network variants are considered here: All-Text, Fuse-Early, Fuse-Late (see Figure 1). Across our datasets, Table 3 shows that the Fuse-Late strategy outperforms the other options for producing predictions from multimodal inputs using a single neural network (including Text-Net). We thus fix this model as our Multimodal-Net used in subsequent experiments.

### Aggregating Transformers and Tabular Models
Having identified a good neural network architecture for multimodal text/tabular inputs, we now study combinations of such models with classical learning algorithms for tabular data. Where not specified, the tabular models are those trained by AutoGluon-Tabular (see Appendix A.5). Here we considered the following aggregation strategies:

- Pre-Embedding, Text-Embedding, Multimodal-Embedding, Weighted-Ensemble, Stack-Ensemble.

The third section of Table 3 illustrates that Stack-Ensemble is overall the best aggregation strategy. As expected, Text-Embedding and Multimodal-Embedding outperform Pre-Embedding, demonstrating how domain-specific fine-tuning improves the quality of learned embeddings. Multimodal-Embedding performs better than Text-Embedding on some datasets and similarly across the rest, showing it can be beneficial to use text representations contextualized on numeric/categorical information.

### AutoGluon Baselines
As most of our results are based around the tabular models in AutoGluon [15], we also compare different variants of AutoGluon (without our Multimodal-Net) as baselines:

- AG-Weighted / AG-Stack: We train AutoGluon with weighted / stack ensemble of its tabular models, here ignoring all text columns.
AG-Weighted + N-Gram / AG-Stack + N-Gram: Similar to AG-Weighted / AG-Stack, except we first use AutoGluon’s N-Gram featurization [14] to encode all text in tabular form. The performance gap between AutoGluon-Tabular with and without N-Grams can reveal (an approximate lower bound for) how much predictive value is provided by the text features in each dataset.

H2O Baselines In addition to AutoGluon, we also run another popular open-source AutoML tool offered by H2O. Since H2O AutoML is not designed for the text in our multimodal data tables, we try combining H2O’s NLP tool [28] and tabular AutoML tool [43].

H2O AutoML: We run H2O AutoML directly on the original data of our benchmark. It is assumed that H2O AutoML ignores all text features (as a tabular AutoML framework), but H2O categorizes text vs. other feature types slightly differently than us. For fair assessment, our benchmark leaves key decisions like training/validation splits and how to designate feature types up to each AutoML tool.

H2O AutoML + Word2Vec: We run H2O’s word2vec algorithm to featurize text fields and then H2O AutoML on the featurized data, following their recommended procedure [28].

H2O AutoML + Pre-Embedding: We featurize each text field using embeddings from a pretrained ELECTRA Transformer, as in Pre-Embedding, followed by H2O AutoML on the featurized data table.

The last section of Table 3 shows that while these powerful AutoML ensemble predictors can outperform our individual neural network models (particularly for datasets with more tabular-signal), our proposed Stack-Ensemble and Weighted-Ensemble are superior overall. Given the success of pretrained Transformers across NLP, we are surprised to find both N-Grams and word2vec here provide superior text featurization than Pre-Embedding.

Performance in Real-world ML Competitions Some datasets in our multimodal benchmark originally stem from ML competitions. For these (and other recent competitions with text/tabular data), we fit our automated solution using the official competition dataset, without manual adjustment or data preprocessing. We then submit its resulting predictions on the competition test data to be scored, which enables us to see how they fare against the manual efforts of human data science teams.

Our Stack-Ensemble model achieves 1st place in two prediction competitions from MachineHack: Product Sentiment Classification⁷ and Predict the Data Scientists Salary in India⁸, and this model achieves 2nd place in another MachineHack competition: Predict the Price of Books⁹, as well as a Kaggle competition: California House Prices⁶. Simply training only our Multimodal-Net suffices to achieve 2nd place in a very popular Kaggle competition in which 2380 teams participated: Mercari Price Suggestion Challenge⁷ (which offered a $100,000 prize). These results demonstrate that, without any manual adjustment, the AutoML strategy identified from our benchmark outperforms top data scientists on real-world text/tabular datasets that possess great commercial value.

Feature Importance Analysis Feature importance helps us understand what drives a ML system’s accuracy and whether text fields in a dataset are worth their overhead. We compute permutation feature importance [5] for three models: the AG-Stack+N-Gram baseline, our Multimodal-Net, and our Stack-Ensemble (containing the Multimodal-Net). The importance of a feature is defined as the drop in prediction accuracy after values of only this feature (which are entire text fields for a text column) are shuffled in the test data (across rows). We only shuffle original column values so our importance scores are not biased by preprocessing/featurization decisions (except in how these directly affect model accuracy). Figure 3 shows that both our Multimodal-Net and Stack-Ensemble with this network rely more heavily on text features than the N-Gram baseline. With more powerful modeling of text fields, models begin to rely more heavily on the text fields. An exception here is the brand_name feature in mercari, but this feature usually contains just a single word in its fields.

---

8https://machinehack.com/hackathons/predict_the_data_scientists_salary_in_india/hackathon/overview (“Xingjian Shi” entry)
9https://machinehack.com/hackathons/predict_the_price_of_books/overview
6https://www.kaggle.com/c/california-house-prices (“sxjscience” entry)
Lacking public benchmarks, academic research on ML for multimodal text/tabular data has not matched industry demand to derive practical value from such data. This paper provides evidence that generic best practices for such data remain unclear today: we simply evaluated a few basic strategies on our benchmark and found a single automated method that turns out to outperform top human data scientists in numerous prediction competitions involving diverse text/tabular data. These results also support the premise that our benchmark is sufficiently diverse and representative of real-world text/tabular prediction tasks. Our rigorous empirical benchmark challenges conventional beliefs:

- Neural embedding of text followed by tabular modeling [4, 28] is in fact often outperformed by simple N-gram featurization or leveraging text neural networks for their predictions (stack ensembling) rather than their representations (embeddings).
- In the architecture of multimodal networks for classification/regression, newer ideas to fuse modalities in early layers (i.e. Transformers with cross-modality attention [32, 51, 54]) are not necessarily superior to older multi-tower architectures that fuse representations in late layers [2, 36, 52].
- A fully end-to-end multimodal neural network approach is much improved by stack ensembling this network with tabular models trained in separate stages rather than end-to-end.

Previously anticipated conclusions that are empirically validated by our benchmark include:

- Fine-tuned networks produce better text featurization (embeddings) than pretrained counterparts. A fine-tuned multimodal network is generally superior, even when only used to featurize text, as its embeddings benefit from being contextualized on the tabular features.
- Naively casting numeric/categorical features as strings (text) is simple yet effective [51].
- ELECTRA appears to be a better model for supervised learning with text than RoBERTa, and additional tricks of exponential decay and checkpoint-averaging boost its performance.
- Able to exploit predictive interactions between different modalities, stack ensembling outperforms simple weighted ensembling, yet it still facilitates modular system design.

Further analysis of our benchmark can reveal many more practical ML insights. For instance, important questions we have not considered here include how to best: Handle many long text fields? Perform multimodal feature selection? Apply feature engineering that combines synergistically with learned neural network representations? Further pretrain networks on the same training data provided for supervised learning? Allocate limited training/HPO time between cheaper tabular models and more expensive text neural networks?

We consider the study presented in this paper as a starting point for multimodal AutoML with text/tabular data. The associated benchmark and methods are open source and we welcome contributions to improve them further. We hope these stimulate the AutoML community to broaden the applicability of their methods to more heterogeneous data types, especially those modalities that commonly co-occur in real-world ML applications. To ensure similar advancements are made for text/tabular data with low-resource languages [31, 37, 40], we encourage the development of a similar benchmark with non-English text. We also caution that analysis of text fields raises particular privacy concerns as such fields tend to be unstructured and may expose arbitrary personal information [6, 19].
References


Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Described limitations of the benchmark and questions our empirical study did not answer.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Discussion section.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See the provided Github repository.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See the Appendix and the provided Github repository.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] See Appendix: Given a limited compute budget, we believe more meaningful conclusions may be drawn by running more algorithms over more datasets rather than replicate runs of different seeds/splits on just a few (less diverse) datasets. We also omitted small datasets from our benchmark for which replicate runs would otherwise be required to get stable results.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Provided links and proper attribution.
   (b) Did you mention the license of the assets? [Yes] Linked in the provided Github repository.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We provided Github repository of our benchmark datasets, code to recreate these datasets from their original data sources, and code to run the methods described in the paper on our benchmark.
   (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [Yes] The datasets that contain data from people all stem from commercial sources where people upload their data intentionally to share it with the world (e.g. user reviews, Kickstarter fundraising, public questions, etc.). There is no sensitive/personal information in these data, beyond what a person intended to publicize. Given we are not the original curators of these datasets, we cannot check with the people whose data they contain, but we are confident these people consent to the information being public (as long as the content license is respected).
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] The data were all already publicly available in different forms, and are not offensive.

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]