FeatureInsight: Visual Support for Error-Driven Feature Ideation in Text Classification

Michael Brooks*, Saleema Amershi†, Bongshin Lee†, Steven M. Drucker†, Ashish Kapoor†, Patrice Simard†

*University of Washington, †Microsoft Research

ABSTRACT

Machine learning requires an effective combination of data, features, and algorithms. While many tools exist for working with machine learning data and algorithms, support for thinking of new features, or *feature ideation*, remains poor. In this paper, we investigate two general approaches to support feature ideation: visual summaries and sets of errors. We present FeatureInsight, an interactive visual analytics tool for building new dictionary features (semantically related groups of words) for text classification problems. FeatureInsight supports an error-driven feature ideation process and provides interactive visual summaries of sets of misclassified documents. We conducted a controlled experiment evaluating both visual summaries and sets of errors in FeatureInsight. Our results show that visual summaries significantly improve feature ideation, especially in combination with sets of errors. Users preferred visual summaries over viewing raw data, and only preferred examining sets when visual summaries were provided. We discuss extensions of both approaches to data types other than text, and point to areas for future research.

Keywords: Features; machine learning; interactive machine learning; text classification; dictionaries; visualization.

Index Terms: Human-centered computing~Visual analytics. Computing methodologies~Machine learning.

1 INTRODUCTION

Machine learning (ML) is increasingly relied upon for applications such as data analysis, intelligent systems, and large-scale computation. Tools and best practices exist to help practitioners with many aspects of creating ML systems, but so far there is little support for a crucial step in the process: thinking of new *features*.

Features are observable quantities for describing data to an ML algorithm (e.g., web pages could be represented by the words, length, or page layout, while images could be represented by hue, saturation, or shapes in the image). Features must capture salient aspects of the data to help the machine learn the target concept (e.g., that a web page *is a sports page* or that an image *contains a car*). But features can hurt performance if they insufficiently describe the concept, capture irrelevant aspects of the data, or are redundant. Although there are many requirements for ML systems (e.g., high quality data, appropriate model choice, parameter tuning), both researchers [1], [2], [3], [4] and practitioners [5], [6] recognize feature engineering as crucial to the machine learning process.

Features for ML systems can have various origins. In some cases, features appropriate for an ML problem can be obtained from related literature [7]. On problems where data is plentiful, features can sometimes be automatically generated [8], [9], [10] or selected from a candidate set [2], [4], [11], [12]. However, when training data is limited (e.g., for personalized tasks [13] or when domain expertise is required and prohibitively expensive to obtain [14]), automatically generated features can result in over-fitting [1]; in these scenarios, human involvement in feature engineering can improve performance [15], [16]. Even with the availability of existing features or automated techniques, practitioners spend extensive time thinking of and developing custom features to support their specific problems [1], [7]. Custom features are often developed in a trial-and-error manner [1], [7] because, aside from simply analyzing classifier errors, there are few recommendations available for how to think of good features, and tool support for *feature ideation* remains underexplored.

In this paper, we develop a hybrid approach to feature ideation that falls between fully human-driven and fully automated feature engineering. We leverage the machine's ability to efficiently analyze and visually summarize a large number of examples, while drawing on human knowledge and intuition to identify and build potentially useful features. We consider two techniques for feature ideation: (1) examining *sets of errors*, to improve feature generalizability; and (2) *visual summaries* of errors to reduce noise and cognitive load. These techniques are implemented in FeatureInsight, a tool for error-driven feature ideation in binary (two-class) classification problems with limited amounts of training data. We focus on text classification problems and word-based features. FeatureInsight visually summarizes sets of misclassified text documents via ranked and annotated lists of promising seed words.

Our main contributions are:

- A design exploration and implementation of two approaches for supporting feature ideation: *sets of errors* and *visual summaries*, in the FeatureInsight system.
- A controlled experiment evaluating the effects of both *sets of errors* and *visual summaries*. We find that visual summaries helped participants create better-performing classifiers and were preferred by users, while examining sets of errors had no benefits without visual summaries.
- An exploratory analysis comparing features from FeatureInsight to automatic feature selection algorithms, illustrating strengths and weaknesses of both methods.

2 BACKGROUND AND RELATED WORK

In this section, we consider the challenge of creating new features and review research on user interaction with ML systems. We then examine related work in text classification.

2.1 Feature Engineering in Machine Learning

The importance of features to ML is well-established [1], [2], [5], [6]. Practitioners often begin by searching for existing features from related problems, using their own domain knowledge to think of relevant features, or trying to algorithmically generate features for the given dataset [1], [7]. However, even with an initial set of

* mjbrooks@uw.edu, †{samershi, bongshin, sdrucker, akapoor, patrice}@microsoft.com

candidate features, practitioners report spending a large amount of time iteratively experimenting with new features [1], [7].

Algorithms have been created to automatically build high-level features out of low-level elements for speech [10], real-time sensors [8], and general ML problems [9]. If a set of candidate features is available, *feature selection* algorithms can find a useful subset of those features for improving efficiency and generalization [4]. For text classification, the most commonly used family of feature selection techniques is metric-based ranking followed by selection of the *k* best features [2], [11], [12], but there is no single metric that is consistently optimal across data sets. In our paper, we perform an exploratory analysis comparing human-generated features to features selected using Information Gain and Chi-Squared metrics, two commonly-used baselines.

When algorithmic techniques are not effective (as in small-dataset problems), custom features must be created manually; and, even when algorithmic feature selection are used, manual feature engineering is still common. However, there is little guidance about how this should be done. One strategy for thinking of new features is *examination of classifier errors* [5], [6]. Here, the practitioner considers misclassified data to identify information the classifier may be missing and explain the error. Features are added to capture the missing information and improve the classifier's performance.

Another approach is to *compare and contrast* examples. Patel *et al.* proposed comparing example data from a classifier's "confused region" with a representative example from the opposite class to help users think of new discriminating features [17]. Stumpf *et al.* used a compare and contrast approach to help users give open-ended feedback to an email classification system [18]. Cheng and Bernstein also used a compare and contrast approach to help crowd workers generate features [16]. We incorporate *error examination* and *compare and contrast* into the FeatureInsight system.

Humans possess knowledge that can be useful to ML systems, but apart from users providing labels, researchers have only recently begun exploring richer forms of user input. While the literature includes examples of visual, interactive data classification [19], [20], few systems help users directly create *features*. Still, there is evidence of the potential for human-driven feature engineering; Stumpf *et al.* showed that free-form feedback on features could improve an ML system [13]. Others have explored enabling users to adjust feature weights [15], [21], interactively debugging features [22], and crowd-generating features [16]. In this paper, we investigate the potential of interactive feature ideation.

2.2 Text Classification

Text classification, our focus in this paper, has important practical applications (e.g., search ranking, information extraction, natural language processing) [23]. Although we use web pages here, our strategies should translate to many other types of text-based data.

A common feature engineering approach for text is "bag-of-words" or *n*-grams [24], [25]. This involves automatically generating a very large feature set where each feature corresponds to the presence or frequency of a specific word or *n*-word phrase in a document. While bag-of-words features are common, they produce large, sparse feature spaces, which typically require more training data to mitigate over-fitting [1]. Because bag-of-words features require no human input, they afford no opportunity to incorporate the users' domain knowledge. In FeatureInsight, we leverage bag-of-words to generate candidate words, which are ranked and displayed to prompt ideas for new features.

Users of FeatureInsight interactively build *dictionary features*, which are groups of semantically related words or *n*-grams. For example, for a *bicycling* web page classifier, a useful dictionary feature may be *types of bicycles* (including words such as "bicycle" or "tricycle") or *bicycling events* (including "bike race" and "bike tour"). Manually-constructed dictionaries were part of early rule-based text classification systems [26]. Recent work uses lexicons or gazetteers built from pre-existing dictionaries or seed words [27], [28], but manually-constructed dictionaries are still used [29]. Dictionary features are relatively easy for users to understand, capture the users' domain knowledge, and reduce the number of features by capturing the semantic equivalence of their constituent words. Using fewer features typically means less training data is needed and generalization is improved. Text and dictionary features also enable our techniques for supporting feature ideation. For example, text is readily decomposed into small units (i.e., words) that can be displayed to the user for semantic grouping into meaningful features (i.e., dictionaries). In the Discussion, we consider extensions to other types of data and features.

3 Supporting Feature Ideation

FeatureInsight helps users think of new dictionary features for binary text classification by displaying interactive visual summaries of sets of errors.

The system ingests a set of text documents, which must be labeled positive or negative for the target class (e.g., *bicycling* or *not-bicycling*). Once this training data is loaded, the FeatureInsight user interface shows a featuring area on the left, and controls for exploring the data on the right (Figure 1). As the user interacts with FeatureInsight, they can add new features in the featuring area (Figure 1, A) and modify or delete features.

When a feature changes, FeatureInsight re-trains a *logistic regression* classifier. This involves checking if the document contains *any* of the words in the dictionary (feature), and calculating the word's term frequency-inverse document frequency (*tf-idf*), a common technique from information retrieval [30]. For the datasets we tested, with a few hundred examples each, classifier training is fast and can be done interactively. To help users track progress, in the top-right corner FeatureInsight shows overall accuracy (percent of training data classified correctly) and change in accuracy since the last re-training.

3.1 Feature Ideation by Comparing Errors

The feature ideation workflow is based on two recommended approaches: examination of classifier errors [5, p. 226], [6], and comparison [13], [16], [17]. With every iteration of classifier training, FeatureInsight lets users examine documents currently being misclassifying. Binary classifiers can make two types of errors: false positives (documents misclassified as positive when actually negative) and false negatives (documents misclassified as negative when actually positive). False positive and negative errors may require different features to correct. Therefore, users can decide which errors to inspect, using controls at the top (Figure 1, B). To help prioritize which errors to work on, the system displays accuracies for the documents within each class. For example, if a classifier is predicting 90% of negative documents correctly but only 40% of positive documents correctly, the user might prioritize fixing errors in the positive class.

FeatureInsight provides side-by-side comparison to highlight characteristics that may help discriminate between positive and negative documents. Misclassified documents of the focused class are shown beside documents of the opposite class (false positives are shown besides positives and false negatives are shown besides negatives). We refer to *misclassified* documents as **errors** and comparison documents for those errors as **contrasts**. When a user switches focus between false positives and false negatives, the corresponding errors are shown on the left, while contrasts are shown on the right. For example, when building a *bicycling* web

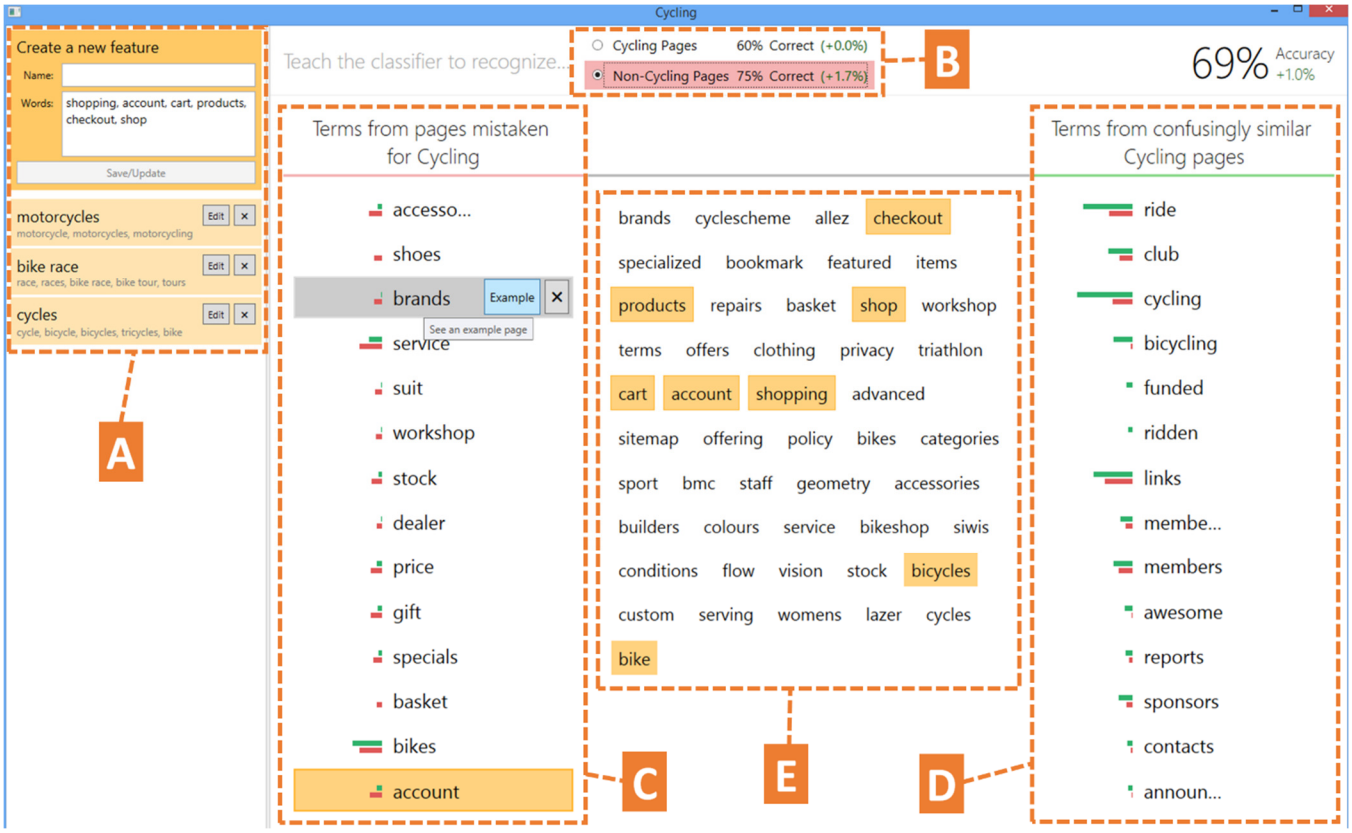

Figure 1. A screenshot of FeatureInsight being used to train a classifier for bicycling web pages: A) the featuring area; B) focus selection controls; C) key words for Errors; D) key words for Contrasts; E) words related to the selected word (*brands*).

page classifier, users could examine *motorcycling* documents misclassified as bicycling, compare them to actual bicycling documents, and try to think of new features to help the classifier distinguish between motorcycling and bicycling pages.

3.2 Examining Sets of Errors with Visual Summaries

Inspecting errors is a recommended way to think of new features [5], [6], but individual errors can yield feature ideas that do not generalize. For example, if our *bicycling* classifier mislabels a "motorcycle club" web page as positive, inspecting this error alone might prompt the user to create a feature from the club's name. While this might correct one error, it would not eliminate confusion about motorcycle clubs in general. Instead, ML practitioners might inspect multiple errors to think of features, a practice we refer to as *examining sets of errors*. Commonalities across many errors may yield feature ideas that generalize beyond the errors considered. In FeatureInsight, the user examines many errors at once.

While examining sets of errors might improve the generalizability of feature ideas, it also increases cognitive load, particularly with unstructured and noisy data like web pages. FeatureInsight uses *visual summaries* to help abstract away unnecessary detail and highlight important characteristics of documents. For text data, possible visual summaries range from complex techniques, as in visualizations of topic models [31], to simpler techniques like "word clouds" [32], [33]. FeatureInsight uses ranked and annotated lists of words (Figure 1, C and D) and semantically related words (Figure 1, E) to visually summarize documents. In the next section we describe how we generate these summaries and how users interact them to build dictionaries.

3.3 Building Dictionaries from Suggested Words

FeatureInsight visually summarizes sets of errors via ranked and annotated lists of words. One word list is used to summarize error documents (Figure 1, C) while another summarizes corresponding contrasts (Figure 1, D). Words are chosen and ranked based on their potential to improve classifier performance as follows:

1. Obtain the frequency of all words occurring in any of the Error and Contrast documents, excluding stop-words and words with less than three characters.
2. As candidate words for Errors, select 100 words which occur more often in the Errors than Contrasts. Also, as candidate words for Contrasts, select 100 words which occur more often in the Contrasts than Errors.
3. For each of the 200 candidate words, evaluate the improvement in logarithmic loss (log-loss, a commonly-used performance metric) obtained if the word by itself were added as a new feature (in addition to any existing features). This helps prevent feature overlap because each word is considered in conjunction with the current set of features.
4. Rank both lists of candidate words by log-loss improvement and select the top 20 from each list to display as key words.

This procedure takes a fraction of a second on the datasets used below. Filtering by frequency difference (Step 2) reduces the words used in the more expensive log-loss computation.

FeatureInsight shows green and red bars next to each word, indicating its frequency in the positive and negative documents, respectively. High frequency words (with larger bars) may generalize better, while words with large differences in bar length

might better discriminate positives and negatives. An "Example" button beside each word can display a document with a high *tf-idf* value for that word. Users may also manually "hide" words they deem unimportant, revealing new words from the ranked lists.

Users might be tempted to create single-word dictionaries, but higher-level features with multiple words are often more powerful. For example, instead of creating individual features for each bicycle part (e.g., tire, chain, pedal, gear, etc.), a "bicycle parts" dictionary would indicate that these words are related, and reduce the dimensionality of the feature space. Higher-level features can also be more interpretable. The word "chain" may occur in multiple contexts, e.g. jewelry or chemistry (as in a "chain reaction"). However, "chain" as part of a dictionary feature that includes the words "tire" and "brake" provides context that "chain" is meant to refer to a bicycle part. To encourage features combining several semantically-related words, FeatureInsight displays a set of "related words" (Figure 1, E). When a user clicks on a word from either the Errors or Contrasts lists, FeatureInsight finds the 25 most closely related words. "Relatedness" is judged based on the cosine similarity between the words' document-vectors: a vector consisting of the word's *tf-idf* values for each document. The 25 words closest to the selected word are displayed in the interface.

Words anywhere in FeatureInsight can be added to a feature by pressing the Control key and clicking on the word (e.g., from the key words, related words, or in example documents). If users think of other words while exploring FeatureInsight's visual summaries, they may manually type their own words into features. When a word is already part of an existing dictionary, it is given a light-orange background in the Error and Contrast lists and related words list, to minimize redundant features.

4 Evaluation

We conducted a controlled experiment to examine FeatureInsight's support for feature ideation using a 2 (*individual* vs. *sets of errors*) x 2 (*raw data* vs. *visual summaries*) within-subjects design.

4.1 Treatments

We built four versions of our tool to test examining individual vs. sets of errors, and raw data vs. visual summaries: *Pair Inspection* (individual errors + raw data), *Set Inspection* (sets of errors + raw data), *Pair Visualization* (individual errors + visual summaries), and *Set Visualization* (sets of errors + visual summaries). FeatureInsight, as described above, represents the *Set Visualization* version. The others are described below.

As with FeatureInsight (Figure 1), the *Pair Visualization* interface (Figure 2, top) shows lists of words for Errors and Contrasts. However, here words are generated from a single error and contrast document rather than from multiple documents. We also removed the word frequency bars, which do not make sense for individual documents. Errors and contrasts are displayed singly; users page through documents with navigation buttons at the top ("Previous" and "Next"). When the user moves to a new error document, contrast documents are automatically resorted by decreasing cosine similarity to the error (in bag-of-words space) to increase the chances that comparing the two documents will prompt good feature ideas. Users can also view the full web page that the words came from by clicking the "Show Page" button.

The *Set Inspection* interface (Figure 2, middle) shows sets of errors and contrasts as raw pages (without visual summaries), in scrollable, zoomable panels. The *Pair Inspection* version (Figure 2, bottom) only shows one error and one contrast web page. As with the *Pair Visualization* version, *Pair Inspection* provides navigation buttons. In both *Inspection* versions, users can add words to the feature's word list by pressing the Control key and clicking a word

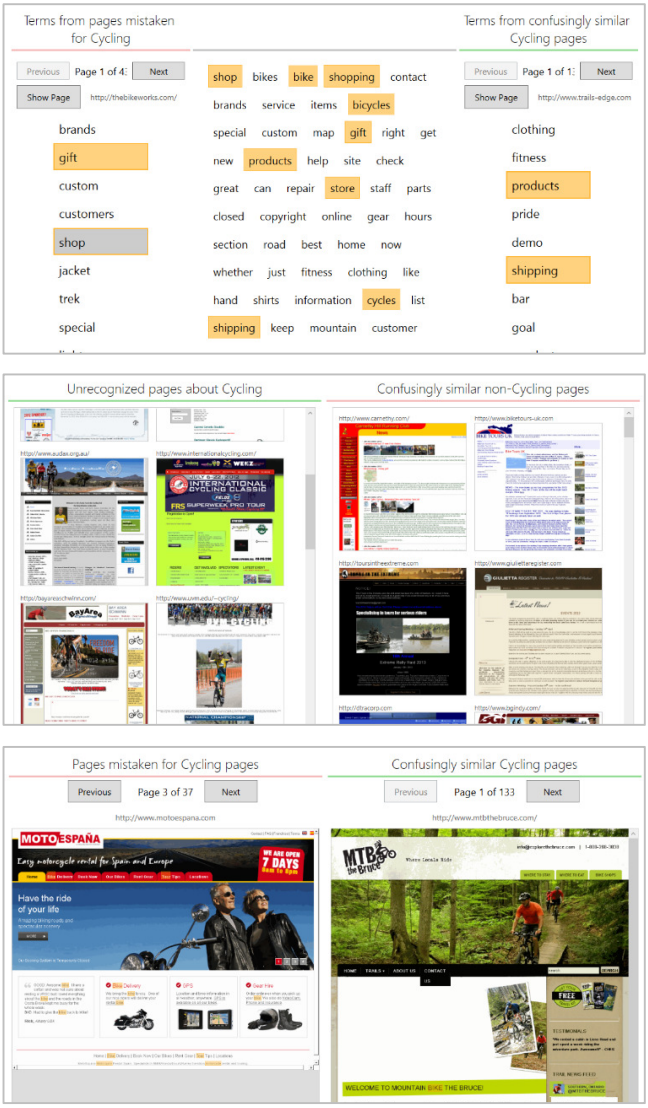

Figure 2: The main content areas for the Pair Visualization (top), Set Inspection (middle), and Pair Inspection (bottom) versions of the FeatureInsight.

in the document. The *Pair Inspection* version can be considered a baseline as it provides support comparable to existing tools.

4.2 The Task and Datasets

Our evaluation focused on the feature ideation process, so we provided participants with already-labeled datasets and then asked them to think of features to build a classifier for a target concept.

Several factors impact the difficulty of feature ideation, including familiarity with the target concept, concept complexity (e.g., class separability), and data ambiguity. In an attempt to control for some of these factors, we used the following approach to create our 8 test datasets for this experiment:

1. We created candidate sets from the Open Directory Project, a community-driven directory of human-categorized web pages (http://www.dmoz.org). We used second-level categories; the top level categories often contain dissimilar subtopics.
2. For each of ~85 categories, we used uncertainty-based sampling to select pages within (positive) and outside (negative) the target concept. Uncertainty-sampling selects

data where a classifier is most uncertain. Intuitively, this finds examples around the decision boundary of the target class, and helps to control for example ambiguity across datasets. We started with a random set of 50 pages, and then iteratively train a classifier (with bag-of-words features) and added pages via sampling until reaching 650 examples.

3. For each candidate dataset, we trained a classifier on a random subset and estimated class separability via the logarithmic loss (log-loss) metric on the remaining test subset. We selected eight datasets of moderate separability (i.e., log-loss close to the median) to help control for concept complexity.
4. Finally, we manually reviewed each dataset, verifying the labels and removing miscategorized pages. This sometimes tipped the class distribution towards positive or negative, so we rebalanced and resized the datasets to achieve 50% positives in each, by randomly discarding pages.

Our datasets were: Equestrianism, Healthcare Business, Sports Shopping, Chemical Business, Museums, Electronics/Electrical, Food Business, and Newspapers.

4.3 Procedure

Each session started with an introduction to the general feature engineering task, the idea and importance of generalizability, and the study procedure. We informed participants that they would be given labeled data, and that their goal was to create features to train a classifier with a high accuracy.

At the start of each condition, we gave participants a short tutorial for the next interface. We let them practice adding features with the *Bicycling* dataset until they were comfortable with the interface. We then gave two five-minute feature ideation tasks in succession. Each task involved a different dataset and target concept (listed above). We did not expect carryover effects between datasets, so we fixed the order of datasets and only counter-balanced the four interface conditions, using a complete set of mutually-orthogonal Latin squares balanced for carryover effects [34]. We expected to observe differences between dataset topics, reflecting a combination of both topic effects and ordering effects like learning and fatigue. Our counterbalancing scheme controls for these effects and allows a fair comparison of the four treatments.

After each condition, participants completed a questionnaire with Likert-type questions about the tool: enjoyment, satisfaction with features they created, and satisfaction with classification performance. At the end of the study, participants filled out a final questionnaire ranking the four tools. The study lasted about 90 minutes, and participants received a $20 coupon.

4.4 Participants and Apparatus

We recruited 20 participants (12 male, 8 female) employed at a large software company, via an internal mailing list for employees interested in machine learning. We required participants to be fluent in English, not red-green colorblind, and familiar with basic machine learning concepts, e.g. features, true positives, and false positives. Participants included software engineers, researchers, and data scientists. Participants completed the study in pairs, using on a 3.47 GHz Windows 8 desktop machines with 16GB RAM and a 24-inch monitor at 1920x1200 resolution.

5 Results

For each task, we evaluated the features participants created on a randomly held-out test set of about 100 documents from the target dataset, based on the area under the precision-recall curve (AUC). AUC gives an estimate of performance that is robust to class imbalance and independent of classification threshold [35]. Unlike log-loss, AUC reflects actual classifier errors. We analyzed the test-set AUC results with a mixed-effects model analysis of variance. Mixed-effects models can be used for factorial designs with within-subjects factors [36], and they treat the experimental subject as a random effect with levels drawn randomly from a population. Our model included fixed effects for *UsingSet* (*Set* vs. *No Set*) and *UsingVis* (*Vis* vs. *No Vis*), as well as the dataset *Topic* (1-8), and the interactions among these. Participant was modeled as a random effect. Note that because the order of dataset topics was fixed, the *Topic* variable reflects topic as well as order effects such as learning and fatigue. For many measures, *Topic* had a significant effect (cited where applicable below).

Because exploratory analysis showed *Topic* had an effect in our study, we used the same mixed-effects analysis for our Likert-type questionnaire data as we used for the AUC, above. Although Likert-type responses are best interpreted as ordinal and analyzed using non-parametric statistics, commonly-used non-parametric approaches for within-subjects designs cannot also statistically control for the effect of dataset topics.

5.1 Feature performance

Participants generated features with an average test-set AUC of 0.76 (±0.024) using the Visualization tools, and 0.70 (±0.036) when using the Inspection tools. This difference was significant, $F(1, 65.5) = 10.7$, $p < 0.01$. Dataset *Topic* had a significant effect on test-set AUC, $F(7, 35.9) = 55.7$, $p < 0.01$. There was no significant difference between Sets vs. Pairs, and no significant interactions. The mean test-set AUC for each tool is shown in Table 1.

5.2 Preference

After using the Visualization interfaces (vs. Inspection), users were significantly more satisfied with their classification performance, more satisfied with the features they created, and reported greater enjoyment: respectively, $F(1, 61.7) = 8.4$ $p < 0.01$; $F(1,59) = 8.2$, $p < 0.01$; and $F(1, 53.8) = 8.2$, $p < 0.01$. For these three measures, there was no significant effect of working with Sets vs. Pairs. For satisfaction with features, dataset *Topic* had a significant effect ($p = 0.049$). For classifier performance, satisfaction with features, and enjoyment, there was also a significant interaction between the *Topic* and Visualization: respectively, $F(3, 30.6) = 3.2$, $p = 0.035$; $F(3,31.7) = 7.0$, $p < 0.01$; $F(3, 34.9) = 3.9$, $p = 0.017$.

The Visualization interfaces were significantly preferred over the Inspection interfaces in the post-study ranking question, $F(1, 63) = 4.9$, $p = 0.031$. Here, the dataset topic also had a significant effect, $F(3, 30.8) = 3.0$, $p = 0.045$. Of the 20 participants, 12 ranked Set Visualization best. Median preference ranks are in Table 1.

5.3 User-Created Dictionaries

We also analyzed the feature sets themselves. Table 1 shows the median number of dictionaries created and the mean number of words per dictionary for each tool. Users of the Set Visualization tool created more dictionaries than with the other tools, but this difference was not significant. The mean number of words per dictionary was not normally distributed, with a skew towards low

Table 1: Test set AUC (under the precision-recall curve) mean and standard deviation, median preference rank, median number of dictionaries, and mean words per dictionary for each treatment.

	AUC	Pref.	Num Dicts.	Words/Dict.
Set Vis	0.78 (0.10)	1	5	6.6
Pair Vis	0.75 (0.12)	2	4	5.7
Set Insp	0.71 (0.12)	3	4	5.5
Pair Insp	0.68 (0.20)	2.5	4	5.5

values, so we applied a natural log transformation. A mixed-effects model analysis found that log-transformed average dictionary size was significantly larger with the Visualization tools vs. the Inspection tools, $F(1, 111) = 10.2$, $p < 0.01$.

6 Discussion and Future Work

Below, we discuss implications for supporting user-driven feature ideation. We supplement our discussion with participants' open-ended responses from the study. We also present a preliminary follow up evaluation to give some insights into the performance of human-generated compared to common automated feature engineering techniques, and discuss areas for future research.

6.1 Visual Summaries Support Feature Ideation

Our study showed that examining visual summaries was significantly preferred and led to better-performing classifiers than examining raw data. Participant responses suggest this may be due to difficulty visually parsing and extracting useful features from raw pages: "*Looking at whole page with images, words as part of image vs. text was slowing me down.*" (P11). P13 also reported: "*Fonts, colors, and images created visual stress.*"

Contrary to expectations, working with sets of errors offered no significant benefits on its own. Instead, examining sets of errors was only useful in combination with visual summaries. Users of the Set Inspection tool explained that their ability to make sense of the raw web pages was reduced even further when multiple pages were presented at once: "*The no-hint/info-overload issue was even worse with many pages at once.*" (P12). One participant said that the Set Inspection tool provided "*too much information 'blasted' at me.*" (P6). Even with multiple errors shown, one participant said that he got fixated on individual errors: "*I found myself picking values for many different features from one source.*" (P3). Simply making *more* misclassified documents available at once may not lead to more effective feature ideation.

Both visual summary tools provided to better classifier performance and preference ratings, but the addition of sets of errors in the Set Visualization tool gave a small improvement over Pair Visualization: "*I also felt like [the Set Visualization] tool got the best accuracy the fastest.*" (P1). Summarizing sets of errors improved the quality of ranked word lists, and may explain why dictionaries created with visual summaries contained significantly more words: "*The tool reminded me of terms, such as HIPAA and EHR, which had a positive impact on accuracy.*" (P19). Several participants also noted that the Pair Visualization tool had noisier words: *"The words shown on each single page do not signal much for the specified topics."* (P8). P19 commented "*More specialized words were discovered in the tool but it is not easy to see impact based on one page at a time.*"

A few participants still preferred to see raw snapshots of web pages instead of visual summaries. They explained that words in isolation lost important context, only available in the original web pages. Of the visual summary tools, P1 said that "*Words lose the meaning without web pages. […] I felt like I was just guessing throughout this part.*" P14 disliked the "*lack of context (web pages) to enable trust in words displayed.*" Context is an important design consideration for supporting feature ideation. Both visual summary tools included buttons to view sample web pages, but future work should consider in-place contextual cues (e.g., showing a few preceding and following words for every word suggested).

6.2 Limitations of Compare and Contrast

FeatureInsight is based on comparing and contrasting misclassified pages with examples from the opposite class, a technique used in previous research [13], [17]. However, several participants commented that the contrasting examples on the right were not very helpful: "*[It is] hard to come up with meaningful groups/features from the right hand side.*" (P18). It was not clear to participants how the contrasts were selected. P41 commented "*It appears that the non-similar page is generated randomly*". Future work should explore showing contrasting examples within some predetermined distance from the error being inspected. In the visual summary conditions, a loss of context may also have hindered effective comparison. Further investigation is necessary to determine the benefits and limits of compare and contrast for feature ideation.

6.3 Debugging and Understanding Features

We designed FeatureInsight to leverage human intuition and domain knowledge with assistance from computational estimates of candidate feature impact. However, some participants stated that they did not have a good understanding of feature quality: "*If an indication can be made in the UI about which features contribute more impact towards accuracy, we can learn to come up with more like them.*" (P2). Automatically estimating feature quality is an interesting open challenge, because features cannot always be considered in isolation [1]; often, features combine with other features in complex ways to affect performance.

In addition to helping users evaluate the quality of features, feature ideation tools could provide guidance on *why* some features are better than others. When building dictionaries it is not clear how broad or specific the concept for the feature should be: is it better to create a feature for *bike chains*, or for all *bike parts*? For *bike repair-shops* or simply *bike repair*? These are subtle distinctions, but they can have a profound impact on feature performance.

6.4 Comparison to Automatic Feature Selection

We ran an exploratory follow-up analysis comparing human-generated features to three different automatic feature selection algorithms to examine the tradeoffs between the two approaches. Note that this analysis reuses features created by participants in our user study, who were under time limits (five minutes). Therefore, while comparing these features to automated techniques may provide insights, it is not necessarily a fair test of the potential competitiveness of human-generated features in general.

With automated techniques, the number of features to create must be specified manually or set through cross validation. In general, the number of features, k, can have an effect on the performance of a feature set, so we automatically generated feature sets for a range of k-values. The three automated techniques we compared our human-generated features to are as follows:

1. *Naïve*. Simulating a user interacting with FeatureInsight. The simulated user's naïve strategy has two steps. First, the highest-ranked word off the Errors list is selected, and this word and the m most closely-related words as assessed by FeatureInsight are added to a new dictionary. Second, the simulated user creates another feature, this time from the Contrasts list. This process is repeated until k features are created. We generated features in this manner for k from 1 to 20 and m set to 1, 5, and 10.
2. *Information Gain.* Each word in the documents is ranked according to its information gain (the decrease in entropy when it is added as a feature) [12]. Then, the k highest-scoring features are selected. We evaluated k set to 5, 10, 15, …, 45.
3. *Chi-Squared*. Words are ranked according to their X^2 statistic, measuring independence between the word and the document class. We used the *scikit-learn* implementation [37]. The top k features were then selected with k set to 5, 10, 15, …, 45.

Using the three techniques above, we generated features for each of the eight training sets in user study. As for our user study, logistic

regression classifiers were trained and evaluated on the test data (roughly 100 examples each). We compared these feature sets to those from the Set Visualization condition of the user study.

Figure 3 illustrates that human-generated features performed similarly to automatic features, in most cases. A mixed-effects model analysis found overall significant differences between the different feature selection techniques, $F(3, 632) = 8.5$, $p < 0.01$; human-generated features had higher AUC than the *Naïve* technique, $p < 0.01$, but no significant difference vs. *Information Gain* and *Chi-Squared*. We analyzed a range of feature set sizes for the automated techniques, and found that the performance appeared to plateau around $k = 10$ for most datasets. Because we lack sufficient data about human-generated features (users generated a median of five features during the 5-minute tasks), it is unclear whether and where the performance of human-generated features also plateaus, or whether performance may continue to improve as time passes. Further investigation is needed to study performance differences between human-generated and automatic feature creation over larger feature sets.

Interestingly, as with user-generated features, the dataset topic had a significant effect on the performance of automated techniques, $F(7,600)=8.8$, $p < 0.001$. Moreover, user-driven feature engineering appears to outperform automated feature generation in some datasets and vice versa. For example user-driven features outperformed automatic techniques on the *Newspapers* dataset by 8-10%; even the participant with the lowest AUC reached 0.90, while the best automatic technique (*Naïve Simulation*) achieved an AUC of only 0.84. In contrast, automatic feature selection outperformed human generated features on the *Museums* dataset, although this difference was not significant. Further research is necessary to understand whether user-driven feature engineering is more useful for some types of data than others.

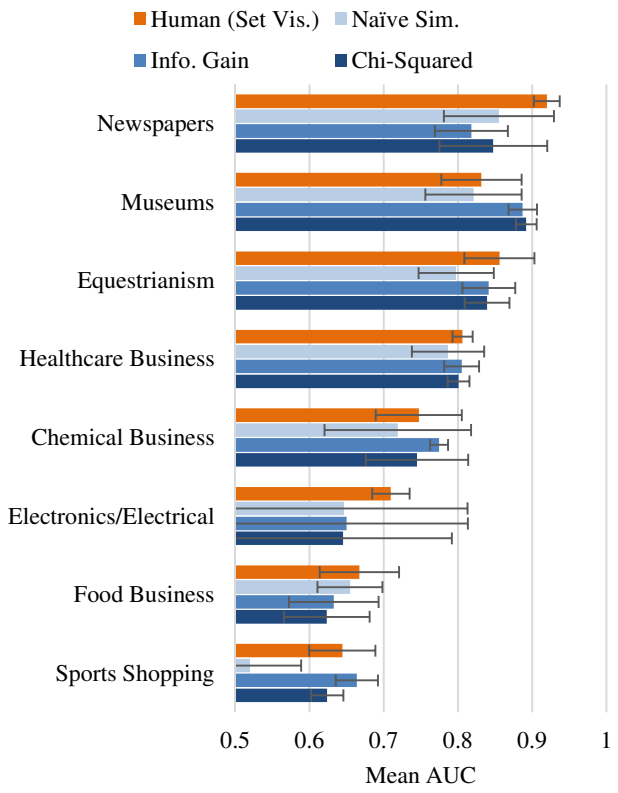

Figure 3: Mean test-set AUC for human-created features and automatic features, separated by dataset topic. Baseline AUC is 0.5. Error bars show std. dev.

So far, feature engineering techniques have been evaluated by their impact on classifier performance. However, feature interpretability also plays a role in practical machine learning applications. As discussed earlier, people spend time iteratively improving the performance of their models, even when starting out with an initial set of features (automatically generated or otherwise). Effective iteration requires hypothesizing about the cause of errors and resolving problems. While previous work has shown that users face difficulty in understanding the rationale and impact behind automatically generated features [18], [21], human-curated features have the potential to be more interpretable. For example, the context provided by semantically grouping and naming features may help users understand the intended meaning of features, which in term may help them debug model performance. Future research should investigate interpretability by, for example, asking practitioners to explain or debug human generated features compared to automatically generated ones.

6.5 Beyond Dictionaries and Text Classification

In this paper, we focused on text classification using dictionary features. The main purpose of FeatureInsight was to support feature ideation, a part of the feature engineering process which we distinguish from feature *implementation*. Dictionaries are a specific way of implementing feature ideas. However, once the user has thought of a new feature idea, other implementations of that idea could be developed, including using regular expressions, rules, or more complex functions. In addition, although FeatureInsight treats words as low-level building blocks that are ranked as seeds for new features, other text features (e.g. capitalization, part of speech, term length, etc.) could be included. For example, in partially structured documents with extracted fields (such as emails), calculated features that compare fields to values could be ranked for different fields, values, and thresholds, and displayed to the user.

We can also consider how the two general approaches of *visual summaries* and *sets of errors* might extend to other types of rich, unstructured data, such as audio, speech, and images. Existing research with these types of data has already produced a large library of meaningful features (e.g., nose and eyebrow detectors, question speech detectors) that can be extracted and used as building blocks for higher-level features. These features are good analogues to the words we have used in FeatureInsight, because they carry human-interpretable meaning relative to the classifier's target concept. The challenge for future research becomes how to visually represent these low-level elements to users in a meaningful way that supports grouping into higher-level semantic features.

7 Conclusion

Machine learning depends on good features, but thinking of new features is difficult. We have explored how visually summarizing sets of errors can provide support for feature ideation by designing, building, and evaluating FeatureInsight, a tool that helps machine learning practitioners interactively define dictionary features for text classification problems. In a controlled experiment evaluating the effects of both *visual summaries* and *sets of errors*, we found that users preferred visual summaries, which led to significantly better classifier performance, while working with sets of errors was only beneficial when combined with visual summaries. These are promising approaches for future tools to support feature ideation both for text classification and beyond.

This paper contributes to the growing area of usable machine learning, wherein human participation in classifier development may produce not only improved machine learning performance, but

also new user interactions and experiences. As machine learning is increasingly a mainstay of modern industrial data science and software development, the potential benefits offered by human-driven interactive machine learning are significant. Visual analytics is one arena in which these ideas are already being discussed. The expansion of this research area represents an opportunity to apply our community's unique perspective, at the intersection of visualization, machine learning, and human-computer interaction, to help make machine learning usable by more people.

ACKNOWLEDGMENTS

We thank the Machine Teaching Group at Microsoft Research for supporting this research, and our reviewers for their insightful feedback in improving the paper.

REFERENCES

[1] P. Domingos, "A few useful things to know about machine learning," *Commun. ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[2] Y. Yang and J. Pedersen, "A comparative study on feature selection in text categorization," presented at the Proceedings of ICML-97, 14th International Conference on Machine Learning, 1997, pp. 412–420.

[3] M. Anderson, D. Antenucci, V. Bittorf, M. Burgess, M. J. Cafarella, A. Kumar, F. Niu, Y. Park, C. Ré, and C. Zhang, "Brainwash: A Data System for Feature Engineering.," in *Proc. CIDR 2013*, 2013.

[4] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artif. Intell.*, vol. 97, no. 1, pp. 245–271, 1997.

[5] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. O'Reilly Media, Inc., 2009.

[6] E. Mayfield and C. P. Rosé, "LightSIDE: Text Mining and Machine Learning User's Manual." 2012.

[7] K. Patel, J. Fogarty, J. A. Landay, and B. Harrison, "Investigating Statistical Machine Learning As a Tool for Software Development," in *Proc. CHI 2008*, New York, NY, USA, 2008.

[8] J. Fogarty and S. E. Hudson, "Toolkit Support for Developing and Deploying Sensor-based Statistical Models of Human Situations," in *Proc. CHI 2007*, New York, NY, USA, 2007.

[9] S. Markovitch and D. Rosenstein, "Feature Generation Using General Constructor Functions," *Mach. Learn.*, vol. 49, no. 1, pp. 59–98, Oct. 2002.

[10] B. Schuller, S. Reiter, and G. Rigoll, "Evolutionary Feature Generation in Speech Emotion Recognition," in *Proc. 2006 Intl. Conf. on Multimedia and Expo*, 2006.

[11] Y. Aphinyanaphongs, L. D. Fu, Z. Li, E. R. Peskin, E. Efstathiadis, C. F. Aliferis, and A. Statnikov, "A comprehensive empirical comparison of modern supervised classification and feature selection methods for text categorization," *J. Assoc. Inf. Sci. Technol.*, vol. 65, no. 10, pp. 1964–1987, 2014.

[12] G. Forman, "An Extensive Empirical Study of Feature Selection Metrics for Text Classification," *J Mach Learn Res*, vol. 3, pp. 1289–1305, Mar. 2003.

[13] S. Stumpf, V. Rajaram, L. Li, W.-K. Wong, M. Burnett, T. Dietterich, E. Sullivan, and J. Herlocker, "Interacting meaningfully with machine learning systems: Three experiments," *IJHCI*, vol. 67, no. 8, pp. 639–662, Aug. 2009.

[14] C. Rosé, Y.-C. Wang, Y. Cui, J. Arguello, K. Stegmann, A. Weinberger, and F. Fischer, "Analyzing collaborative learning processes automatically: Exploiting the advances of computational linguistics in computer-supported collaborative learning," *Comput.-Support. Collab. Learn.*, vol. 3, no. 3, pp. 237–271, Jan. 2008.

[15] H. Raghavan, O. Madani, and R. Jones, "InterActive Feature Selection," in *Proc. IJCAI 2005*, 2005, vol. 5.

[16] J. Cheng and M. S. Bernstein, "Flock: Hybrid Crowd-Machine Learning Classifiers," 2015, pp. 600–611.

[17] K. Patel, S. M. Drucker, J. Fogarty, A. Kapoor, and D. S. Tan, "Using multiple models to understand data," in *Proc. IJCAI 2011*, 2011.

[18] S. Stumpf, V. Rajaram, L. Li, M. Burnett, T. Dietterich, E. Sullivan, R. Drummond, and J. Herlocker, "Toward harnessing user feedback for machine learning," in *Proc. IUI 2007*, 2007.

[19] J. G. S. Paiva, W. R. Schwartz, H. Pedrini, and R. Minghim, "An Approach to Supporting Incremental Visual Data Classification," *IEEE Trans. Vis. Comput. Graph.*, vol. 21, no. 1, pp. 4–17, Jan. 2015.

[20] J. Choo, H. Lee, J. Kihm, and H. Park, "iVisClassifier: An interactive visual analytics system for classification based on supervised dimension reduction," in *2010 IEEE Symposium on Visual Analytics Science and Technology (VAST)*, 2010, pp. 27–34.

[21] T. Kulesza, W.-K. Wong, S. Stumpf, S. Perona, R. White, M. M. Burnett, I. Oberst, and A. J. Ko, "Fixing the Program My Computer Learned: Barriers for End Users, Challenges for the Machine," in *Proc. IUI 2009*, New York, NY, USA, 2009.

[22] F. Heimerl, C. Jochim, S. Koch, and T. Ertl, "FeatureForge: A Novel Tool for Visually Supported Feature Engineering and Corpus Revision," in *Proceedings of COLING 2012: Posters*, Mumbai, India, 2012, pp. 461–470.

[23] F. Sebastiani, "Machine Learning in Automated Text Categorization," *ACM Comput. Surv.*, vol. 34, no. 1, Mar. 2002.

[24] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Proc. ECML 1998*, 1998.

[25] S. Scott and S. Matwin, "Feature engineering for text classification," in *Proc. ICML 1999*, 1999, vol. 99, pp. 379–388.

[26] P. J. Stone and E. B. Hunt, "A Computer Approach to Content Analysis: Studies Using the General Inquirer System," in *Proc. 1963 Joint Computer Conference*, New York, NY, USA, 1963.

[27] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural Language Processing (Almost) from Scratch," *JMLR*, vol. 12, pp. 2493–2537, Nov. 2011.

[28] A. Smith and M. Osborne, "Using Gazetteers in Discriminative Information Extraction," in *Proc. Conf. on Comp. Natural Language Learning 2006*, Stroudsburg, PA, USA, 2006.

[29] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, "Lexicon-Based Methods for Sentiment Analysis," *Comput. Linguist.*, vol. 37, no. 2, pp. 267–307, Apr. 2011.

[30] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manag.*, vol. 24, no. 5, pp. 513–523, 1988.

[31] J. Chuang, C. D. Manning, and J. Heer, "Termite: Visualization Techniques for Assessing Textual Topic Models," in *Proc. AVI 2012*, New York, NY, USA, 2012.

[32] F. B. Viégas and M. Wattenberg, "TIMELINES: Tag Clouds and the Case for Vernacular Visualization," *interactions*, vol. 15, no. 4, pp. 49–52, Jul. 2008.

[33] G. Coppersmith and E. Kelly, "Dynamic Wordclouds and Vennclouds for Exploratory Data Analysis," in *Proc. 2014 ACL Workshop on Interactive Language Learning, Visualization, and Interfaces*, 2014.

[34] E. Williams, "Experimental Designs Balanced for the Estimation of Residual Effects of Treatments," *Aust. J. Chem.*, vol. 2, no. 2, pp. 149–168, Jan. 1949.

[35] J. Davis and M. Goadrich, "The relationship between Precision-Recall and ROC curves," in *Proc. ICML 2006*, 2006.

[36] R. D. Wolfinger, R. D. Tobias, and J. Sall, "Mixed models: a future direction," in *SAS Users Group Conf.*, 1991, pp. 1380–1388.

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.