CrossCheck: Rapid, Reproducible, and Interpretable Model Evaluation

Dustin Arendt* Zhuanyi Shaw*
Prasha Shrestha† Ellyn Ayton† Maria Glenski† Svitlana Volkova*
*Visual Analytics Group, †Data Sciences and Analytics Group
Pacific Northwest National Laboratory
{first}.{last}@pnnl.gov

Abstract

Evaluation beyond aggregate performance metrics, e.g. F1-score, is crucial to both establish an appropriate level of trust in machine learning models and identify avenues for future model improvements. In this paper we demonstrate CrossCheck, an interactive capability for rapid cross-model comparison and reproducible error analysis. We describe the tool, discuss design and implementation details, and present three NLP use cases – named entity recognition, reading comprehension, and clickbait detection that show the benefits of using the tool for model evaluation. CrossCheck enables users to make informed decisions when choosing between multiple models, identify when the models are correct and for which examples, investigate whether the models are making the same mistakes as humans, evaluate models' generalizability and highlight models' limitations, strengths and weaknesses. Furthermore, CrossCheck is implemented as a Jupyter widget, which allows for rapid and convenient integration into existing model development workflows.

1 Motivation

AI models are often imperfect, opaque, and brittle. Gaining actionable insights about model strengths and weaknesses is challenging because simple metrics like accuracy or F1-score are not sufficient to capture the complex relationships between model inputs and outputs. Many researchers agree that ML models have to be optimized not only for expected task performance but for other important criteria such as explainability, interpretability, reliability, and fairness that are prerequisites for trust (Lipton, 2016; Doshi-Velez and Kim, 2017; Poursabzi-Sangdeh et al., 2018). Standard performance metrics can be augmented with exploratory model performance analysis, where a user can interact with inputs and outputs to find patterns or biases in the way the model makes mistakes to answer the questions of when, how, and why the model fails.

To support ML model evaluation beyond standard performance metrics, we developed a novel interactive tool CrossCheck[1]. Unlike several recently developed tools for analyzing model errors (Agarwal et al., 2014; Wu et al., 2019), understanding model outputs (Lee et al., 2019; Hohman et al., 2019), and model interpretation and diagnostics (Kahng et al., 2017, 2016; Zhang et al., 2018), CrossCheck is designed to allow rapid prototyping and cross-model comparison iteratively during model development to support comprehensive experimental setup and gain interpretable and informative insights into model performance.

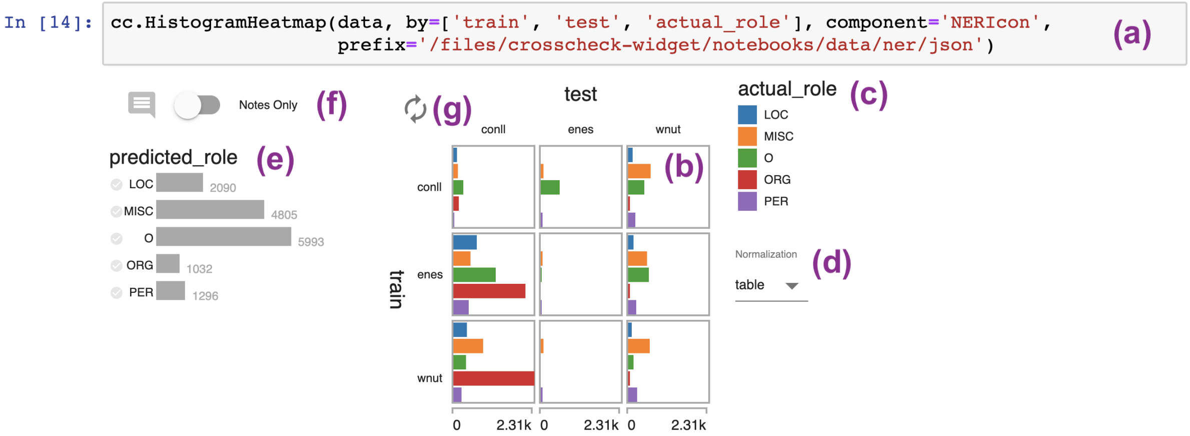
Many visualization tools have been developed recently, *e.g.,* ConvNetJS[2], TensorFlow Playground[3], that focus on structural interpretability (Kulesza et al., 2013; Hoffman et al., 2018) and operate in the neuron activation space to explain models' internal decision making processes (Kahng et al., 2017) or focus on visualizing a model's decision boundary to increase user trust (Ribeiro et al., 2016). Instead, CrossCheck targets functional interpretability and operates in the model output space to diagnose and contrast model performance.

Similar work to CrossCheck includes AllenNLP Interpret (Wallace et al., 2019) and Errudite (Wu et al., 2019). AllenNLP Interpret relies on saliency map visualizations to uncover model biases, find decision rules, and diagnose model errors. Errudite implements a domain specific language for counterfactual explanations. Errudite and AllenNLP Interpret focus primarily on error analysis for a single model, while our tool is specifically designed for contrastive evaluation across multiple models *e.g.,* neural architectures with different parameters.

Manifold (Zhang et al., 2018) supports cross-

[1] https://github.com/pnnl/crosscheck
[2] https://github.com/karpathy/convnetjs
[3] https://playground.tensorflow.org/

Proceedings of the 2nd Workshop on Data Science with Human in the Loop: Language Advances, pages 79–85
June 11, 2021. ©2021 Association for Computational Linguistics



```
In [14]: cc.HistogramHeatmap(data, by=['train', 'test', 'actual_role'], component='NERIcon',
                             prefix='/files/crosscheck-widget/notebooks/data/ner/json')
```

Figure 1: `CrossCheck` embedded in a Jupyter Notebook cell: (a) code used to instantiate the widget (b) the histogram heatmap shows the distribution of the third variable for each combination of the first two (c) the legend for the third variable (d) normalization controls (e) histogram & filter for remaining variables (f) controls for notes (g) button to transpose the rows and columns.

model evaluation, however the tool is narrowly focused on model confidence and errors via pairwise model comparison with scatter plots which is quite limited and does not satisfy the needs of complex NLP tasks. `CrossCheck` enables users to investigate “where” and “what” types of errors models make and, most importantly, assists the user with answering the question of “why" a model makes that error by relying on a set of derived attributes from the input like inter-annotator agreement, question type, answer length, the input paragraph, *etc.*

We built `CrossCheck` to make our existing error analysis workflow faster and reproducible, reducing human effort to replicate exploratory analyses of new models. `CrossCheck` helps calibrate trust by enabling users to:

- contrast multiple models,
- see when the model is right (or wrong), understand the relationship between correctness and confidence, and examine those examples,
- investigate whether the model makes the same mistakes as humans,
- highlight model limitations, and
- understand how models generalize across domains, languages, and datasets – which has pervasive demand across NLP.

2 CrossCheck

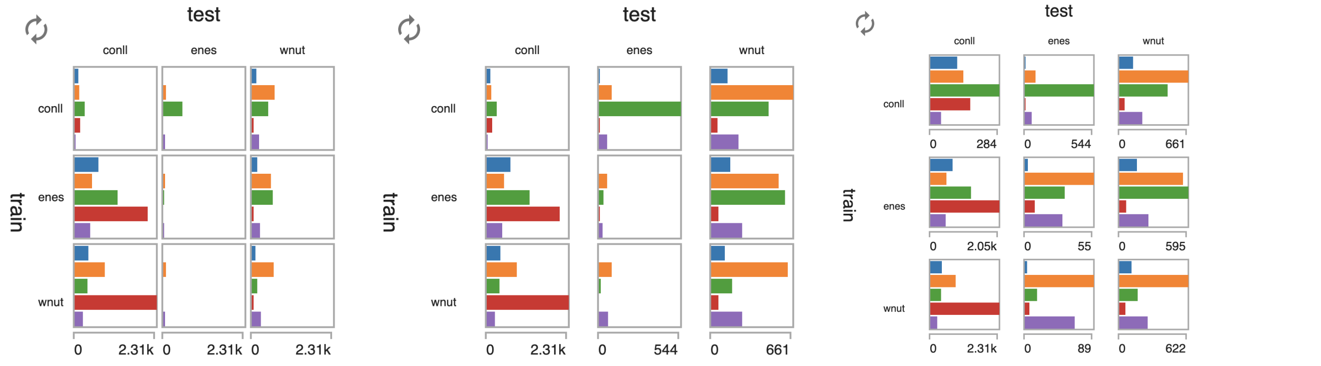
`CrossCheck` is embedded in a Jupyter[4] notebook and input is a single mixed-type table, *i.e.* a pandas DataFrame[5], allowing for tight integration with data scientists’ workflows (see Figure 1a). Below we outline the features of `CrossCheck` in detail.

`CrossCheck`’s main view (see Figure 1b) extends the *confusion matrix* visualization technique by replacing each cell in the matrix with a histogram — we call this view the histogram heatmap. Each cell shows the distribution of a third variable conditioned on the values of the corresponding row and column variables. Every bar represents a subset of instances, *i.e.,* rows in the input table, and encodes the relative size of that group. This view also contains a legend showing the bins or categories for this third variable (see Figure 1c).

The histograms in each cell in `CrossCheck` are drawn horizontally to encourage comparison across cells in the vertical direction. `CrossCheck` supports three normalization schemes (see Figure 1d), *i.e.,* setting the limit of the x-axis in each cell: (1) normalizing by the maximum count within the entire matrix, (2) within each column, or (3) within each cell. We hide certain axes and adjust the padding between cells to emphasize the selected normalization. Figure 2 illustrates how these different normalization options appear in `CrossCheck`. By design, there is no equivalent row normalization option, but the matrix can be transposed (see Figure 1g) for an equivalent effect.

Any variables not directly compared in the histogram heatmap are visualized on the left side of the widget as histograms (see Figure 1e). These histograms also allow the user to filter data when it is rendered in the main view by clicking on the bar(s) corresponding to the data they want to keep.

[4] https://jupyter.org
[5] http://pandas.pydata.org



(a) by table

(b) by column

(c) by cell

Figure 2: CrossCheck supports three histogram normalization options that affect how axes and padding are rendered to improve the readability and interpretation of the view (a) by table: minimal padding, the same x-axes are shown on the bottom row (b) by column: extra padding between columns, different x-axes are shown on the bottom row (c) by cell: extra padding between rows and columns, different x-axes are shown for each cell.

Users can click on any bar in the histogram heatmap to view those instances in a sidebar where they can annotate noteworthy findings. Enabling "Notes Only" (see Figure 1f) shows only instances that have been annotated in the histogram heatmap, revealing what has been annotated in the context of the current variable groupings.

3 Use Cases and Evaluation

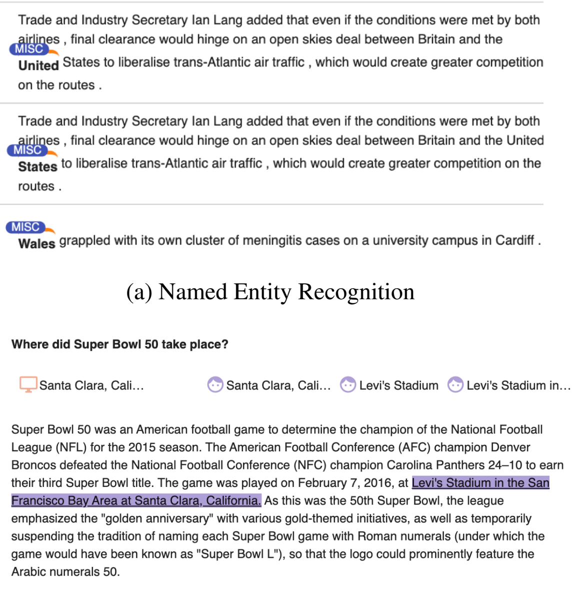
In this section, we highlight how CrossCheck can be used in core NLP tasks such as named entity recognition (NER) and reading comprehension (RC) or practical applications of NLP such as clickbait detection (CB). We present an overview of the datasets used for each task below:

- NER: CoNLL (Sang, 2003), ENES (Aguilar et al., 2018), WNUT 17 Emerging Entities (Derczynski et al., 2017)[6],
- MC: Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016)[7],
- CB: Clickbait Challenge 2017 (Potthast et al., 2018)[8].

3.1 Named Entity Recognition (NER)

To showcase CrossCheck, we trained and evaluated the AllenNLP NER model (Peters et al., 2017) across three benchmark datasets – CoNLL, WNUT, and ENES, producing nine different evaluations. The model output includes, on a per-token level, the model prediction, the ground truth, the original sentence (for context), and what the training and testing datasets were as shown in Figure 3a.

This experiment was designed to let us understand how models trained on different datasets generalize to the same test data (shown in columns) and how models trained on the same training data transfer to perform across different test datasets (shown in rows). Figure 2 illustrates the CrossCheck grid of train versus test datasets. The data has been filtered so that only errors contribute to the bars so we see a distribution of errors per train-test combination across the actual role. The CoNLL dataset is much larger so we normalize within columns in Figure 2b to find patterns within those sub-groups.



Trade and Industry Secretary Ian Lang added that even if the conditions were met by both airlines , final clearance would hinge on an open skies deal between Britain and the MISC **United** States to liberalise trans-Atlantic air traffic , which would create greater competition on the routes .

Trade and Industry Secretary Ian Lang added that even if the conditions were met by both airlines , final clearance would hinge on an open skies deal between Britain and the United MISC **States** to liberalise trans-Atlantic air traffic , which would create greater competition on the routes .

MISC **Wales** grappled with its own cluster of meningitis cases on a university campus in Cardiff .

(a) Named Entity Recognition

Where did Super Bowl 50 take place?

Santa Clara, Cali... Santa Clara, Cali... Levi's Stadium Levi's Stadium in...

Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California. As this was the 50th Super Bowl, the league emphasized the "golden anniversary" with various gold-themed initiatives, as well as temporarily suspending the tradition of naming each Super Bowl game with Roman numerals (under which the game would have been known as "Super Bowl L"), so that the logo could prominently feature the Arabic numerals 50.

(b) Reading Comprehension

Figure 3: Examples of model outputs in CrossCheck for core NLP tasks – for the NER task (above), predicted named entities are highlighted, and for the RC task (below), predicted answer span is highlighted.

[6] github.com/leondz/emerging_entities_17

[7] rajpurkar.github.io/SQuAD-explorer/

[8] www.clickbait-challenge.org/#data

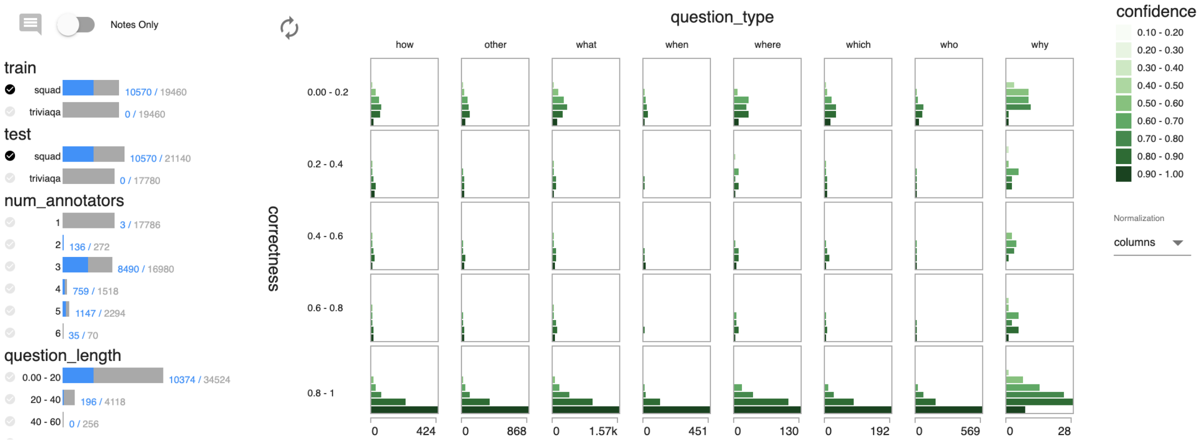


Figure 4: `CrossCheck` for evaluation of reading comprehension models to understand the relationship between correctness, confidence and question types. This highlight models limitations and shows for what examples the model answers correctly.

Table 1: Traditional evaluation: F1-scores for the NER models trained and tested across domains.

Train \ Test	CoNLL	WNUT	ENES
CoNLL	92.51	40.10	11.88
WNUT	55.75	44.73	33.33
ENES	50.78	57.48	64.00

For the same experimental setup, Table 1 summarizes performance with F1-scores. Unlike the F1-score table, `CrossCheck` reveals that models trained on social media data misclassify ORG on the news data, and the news models overpredict named entities on social media data.

3.2 Reading Comprehension (RC)

Similar to NER, we trained an AllenNLP model for reading comprehension (Seo et al., 2016) that is designed to find the most relevant span for a question and paragraph input pair. The model output includes, on a question-paragraph level: the model prediction span, ground truth span, model confidence, question type and length, the number of annotators per question, and what the train and test datasets were, as shown in Figure 3b.[9] Figure 4 contrasts model correctness and confidence across question types. `CrossCheck` reveals that across all types of questions when the model is correct it has higher confidence (bottom row) and lower confidence when incorrect (top row). It also reveals that models have a higher variability in confidence when predicting "why" questions.

3.3 Clickbait Detection

Finally, we demonstrate `CrossCheck` for comparison of regression models. We use a relevant application of NLP in the domain of deception detection (clickbait detection) that was the focus of the Clickbait Challenge 2017, a shared task focused on identifying a score (from 0 to 1) of how "clickbait-y" a social media post (*i.e.,* tweet on Twitter) is, given the content of the post (text and images) and the linked article webpages. We use the validation dataset that contains 19,538 posts (4,761 identified as clickbait) and pre-trained models released on GitHub after the challenge by two teams (*blobfish* and *striped-bass)*[10].

In Figure 5 we illustrate how `CrossCheck` can be used to compare across multiple models and across multiple classes of models.[11] When filtered to show only the *striped-bass* models (shown at right), a strategy to predict coarse (0 or 1) clickbait scores versus fine-grained clickbait scores is clearly evident in the *striped-bass* model predictions. Here, there is a complete lack of predictions falling within the center three columns so even with no filters selected (shown at left), `CrossCheck` indicates that there is an inconsistency in the range of outcomes between models (an explanation for the disparity in F1-scores in Table 2). In cases

[9] We evaluated RC on SQuAD and TriviaQA datasets, but with space limitations only present results for SQuAD.

[10] Models and code were available via `github.com/clickbait-challenge/` repositories.

[11] Note, models could also be grouped by any number of shared characteristics such as the algorithms or architectures used (*e.g.,* different neural architectures used in deep learning models, or models that use deep learning versus those that do not), hyper-parameter settings, granularity of prediction outputs, ensembles versus single models, *etc.*

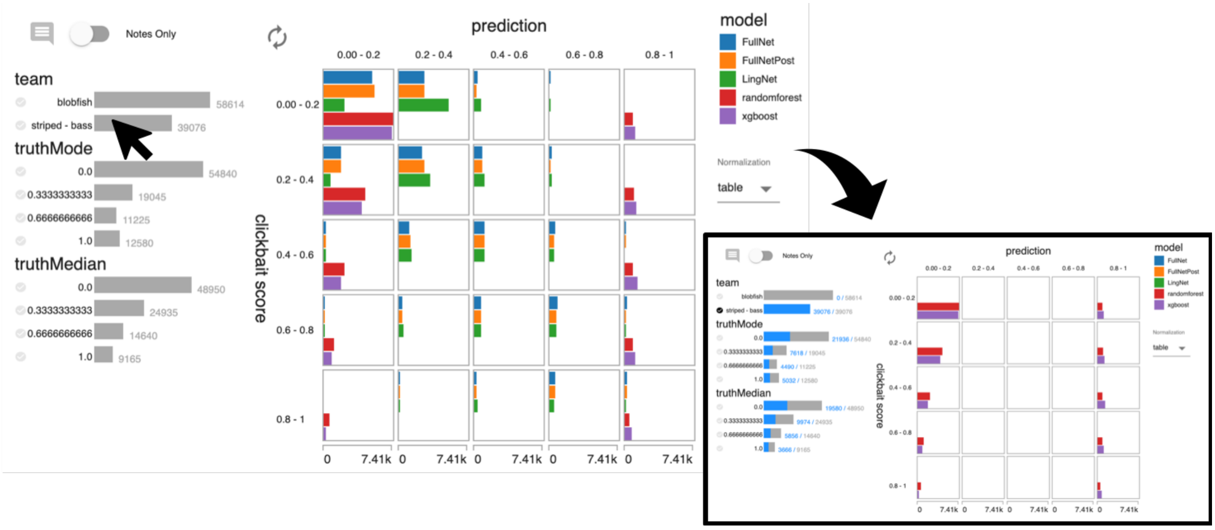


Figure 5: CrossCheck for cross-model comparison across two teams who competed in the Clickbait Challenge 2017 shared task, highlighting distinctions in the variety of prediction outputs with histograms normalized across the full table that become particularly clear when team filters are selected.

Table 2: Traditional evaluation summary table contrasting mean squared error (MSE) and mean absolute error (MAE) of each model's predictions.

Team	Model	MSE	MAE
blobfish	FullNetPost	0.026	0.126
	FullNet	0.027	0.130
	LingNet	0.038	0.157
striped-bass	xgboost	0.171	0.326
	randomforest	0.180	0.336

where there is a more nuanced or subtle disparity, shallow exploration with different filters within CrossCheck can lead to efficient, effective identification of these key differences in functional model behavior.

4 Design and Implementation

We designed CrossCheck following a user-centered design methodology. This is a continuous, iterative process where we identify needs and goals, implement prototypes, and solicit feedback from our users to incorporate in the tool. Our users were data scientists, specifically NLP researchers and practitioners, tasked with the aforementioned model evaluation challenge. We identified CrossCheck's goals as allowing the user to: understand how instance attributes relate to model errors; provide convenient access to raw instance data; integrate into a data scientists workflow; and reveal and understand disagreement across models, and support core NLP tasks and applications.

4.1 Design Iterations

Round 1—Heatmaps (functional prototype) Our first iteration extended the confusion matrix visualization technique with a functional prototype that grouped the data by one variable, and showed a separate heatmap for each distinct value in that group. *User feedback: though heatmaps are familiar, the grouping made the visualization misleading and difficult to learn.*

Round 2—Table & Heatmap (wireframes) We wireframed a standalone tool with histogram filters, a sortable table, and a more traditional heatmap visualization with a rectangular brush to reveal raw instance data. *User feedback: the sortable table and brushing would be useful, but the heatmap has essentially the same limitations as confusion matrices.*

Round 3—Histogram Heatmap (wireframes) We wireframed a modified heatmap where each cell was replaced with a histogram showing the distribution of a third variable conditioned on the row and column variables. This modified heatmap was repeated for each variable in the dataset except for the row and column variables. *User feedback: Putting the histogram inside the heatmap seems useful, but multiple copies would be overwhelming and too small to read. We would prefer to work with just one histogram heatmap.*

Round 4—CrossCheck (functional prototype) We implemented a single "histogram heatmap" inside a Jupyter widget, and made raw instance data available to explore by clicking on any bar. Additionally we incorporated histogram filters from the Round 2 design and allowed the user to change the histogram normalization. *User feedback: the tool was very useful, but could use minor improvements e.g., labeled axes and filtering, as well as ability to add annotation on raw data.*

Round 5—`CrossCheck` (polished prototype) We added minor features like a legend, a matrix transpose button, axis labels, dynamic padding between rows and columns (based on normalization), and the ability to annotate instances with notes. *User feedback: the tool works very well, but screenshots aren't suitable to use in publications.*

4.2 Implementation Challenges

To overcome the rate limit between the python kernel and the web browser (see the `NotebookApp.iopub_data_rate_limit` Jupyter argument) our implementation separates raw instance data from tabular data to be visualized in `CrossCheck`'s histogram heatmap. The tool groups tabular data by each field in the table and passed as a list of each unique field/value combinations and the corresponding instances within that bin. This is computed efficiently within the python kernel (via a pandas `groupby`). This pre-grouping reduces the size of the payload passed from the python kernel to the web browser and allows for the widget to behave more responsively because visualization and filtering routines do not need to iterate over every instance in the dataset. The tool stores raw instance data as individual JSON files on disk in a path visible to the Jupyter notebook environment. When the user clicks to reveal raw instance data, this data is retrieved asynchronously using the web browser's XMLHttpRequest (XHR). This allows the web browser to only retrieve and render the few detailed instances the user is viewing at a time.

5 Discussion

`CrossCheck` is designed to quickly and easily explore many combinations of characteristics of models (*e.g.,* parameter settings, network architectures) as well as datasets used for training or evaluation. It also provides users the ability to efficiently compare and explore model behavior in specific situations and *generalizability of models* across datasets or domains. `CrossCheck` can also generalize to support model comparison, e.g. when ground truth is absent, by visualizing model agreement.

`CrossCheck` enables users to perform error analysis in an efficient, concise, and reproducible manner due to its effective integration into data scientists' workflows. The tool can be used to evaluate across models trained on image, video, tabular data, or combinations of data types with interactive exploration of specific instances on demand. A limitation of the tool is that adding new use cases might require end users to write custom JavaScript code to visualize instances in the details sidebar beyond what is currently implemented. Future work includes expanding to include additional generic components to cover more core NLP or ML tasks.

6 Conclusions

We have presented `CrossCheck`, a new interactive capability that enables rapid model evaluation and error analysis. There are several key benefits to performing evaluation and analyses using CrossCheck, especially compared to *i.e.,* adhoc or manual approaches because `CrossCheck` is generalizable across text, images, video, tabular, or combinations of multiple data types, can be integrated directly into existing workflows for rapid and highly reproducible error analysis during and after model development, users can interactively explore errors conditioning on different model/data features, and users can view specific instances of inputs that cause model errors or other interesting behavior within the tool itself.

References

Apoorv Agarwal, Ankit Agarwal, and Deepak Mittal. 2014. An error analysis tool for natural language processing and applied machine learning. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations*, pages 1–5.

Gustavo Aguilar, Fahad AlGhamdi, Victor Soto, Mona Diab, Julia Hirschberg, and Thamar Solorio. 2018. Named entity recognition on code-switched data: Overview of the calcs 2018 shared task. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 138–147.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the wnut2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 140–147.

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Robert Hoffman, Tim Miller, Shane T Mueller, Gary Klein, and William J Clancey. 2018. Explaining explanation, part 4: a deep dive on deep nets. *IEEE Intelligent Systems*, 33(3):87–95.

Fred Hohman, Andrew Head, Rich Caruana, Robert DeLine, and Steven M Drucker. 2019. Gamut: A design probe to understand how data scientists understand machine learning models. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, page 579. ACM.

Minsuk Kahng, Pierre Y Andrews, Aditya Kalro, and Duen Horng Polo Chau. 2017. ActiVis: Visual exploration of industry-scale deep neural network models. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):88–97.

Minsuk Kahng, Dezhi Fang, and Duen Horng Polo Chau. 2016. Visual exploration of machine learning results using data cube analysis. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, page 1. ACM.

Todd Kulesza, Simone Stumpf, Margaret Burnett, Sherry Yang, Irwin Kwan, and Weng-Keen Wong. 2013. Too much, too little, or just right? ways explanations impact end users' mental models. In *2013 IEEE Symposium on Visual Languages and Human Centric Computing*, pages 3–10. IEEE.

Gyeongbok Lee, Sungdong Kim, and Seung-won Hwang. 2019. Qadiver: Interactive framework for diagnosing qa models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9861–9862.

Zachary C Lipton. 2016. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*.

Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*.

Martin Potthast, Tim Gollub, Kristof Komlossy, Sebastian Schuster, Matti Wiegmann, Erika Patricia Garces Fernandez, Matthias Hagen, and Benno Stein. 2018. Crowdsourcing a large corpus of clickbait on twitter. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1498–1507.

Forough Poursabzi-Sangdeh, Daniel G Goldstein, Jake M Hofman, Jennifer Wortman Vaughan, and Hanna Wallach. 2018. Manipulating and measuring model interpretability. *arXiv preprint arXiv:1802.07810*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM.

Tjong Kim Sang. 2003. De meulder, 2003. tjong kim sang, ef, & de meulder, f.(2003). introduction to the conll-2003 shared task: language-independent named entity recognition. In *Proceedings of the Conference on Computational Natural Language Learning. Edmonton, Canada*, pages 142–147.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Eric Wallace, Jens Tuyls, Junlin Wang, Sanjay Subramanian, Matt Gardner, and Sameer Singh. 2019. Allennlp interpret: A framework for explaining predictions of nlp models. *arXiv preprint arXiv:1909.09251*.

Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel S Weld. 2019. Errudite: Scalable, reproducible, and testable error analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 747–763.

Jiawei Zhang, Yang Wang, Piero Molino, Lezhi Li, and David S Ebert. 2018. Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):364–373.