Outlier Detection for Improved Data Quality and Diversity in Dialog Systems

Stefan Larson **Anish Mahendran** **Andrew Lee** **Jonathan K. Kummerfeld**
Parker Hill **Michael A. Laurenzano** **Johann Hauswald** **Lingjia Tang** **Jason Mars**
Clinc, Inc.
Ann Arbor, MI, USA
{stefan,anish,andrew,jkk}@clinc.com
{parkerhh,mike,johann,lingjia,jason}@clinc.com

Abstract

In a corpus of data, outliers are either *errors*: mistakes in the data that are counterproductive, or are *unique*: informative samples that improve model robustness. Identifying outliers can lead to better datasets by (1) removing noise in datasets and (2) guiding collection of additional data to fill gaps. However, the problem of detecting both outlier types has received relatively little attention in NLP, particularly for dialog systems. We introduce a simple and effective technique for detecting both erroneous and unique samples in a corpus of short texts using neural sentence embeddings combined with distance-based outlier detection. We also present a novel data collection pipeline built atop our detection technique to automatically and iteratively mine unique data samples while discarding erroneous samples. Experiments show that our outlier detection technique is effective at finding errors while our data collection pipeline yields highly diverse corpora that in turn produce more robust intent classification and slot-filling models.

1 Introduction

High-quality annotated data is one of the fundamental drivers of progress in Natural Language Processing (e.g. Marcus et al., 1993; Koehn, 2005). In order to be effective at producing an accurate and robust model, a dataset needs to be correct while also diverse enough to cover the full range of ways in which the phenomena it targets occur. Substantial research effort has considered dataset correctness (Eskin, 2000; Dickinson and Meurers, 2003; Rehbein and Ruppenhofer, 2017), particularly for crowdsourcing (Snow et al., 2008; Jiang et al., 2017), but addressing diversity in data has received less attention, with the exception of using data from diverse domains (Hovy et al., 2006). Outlier detection, the task of finding examples in a dataset that are atypical, provides a means of approaching the questions of correctness and diversity, but has mainly been studied at the document level (Guthrie et al., 2008; Zhuang et al., 2017), whereas texts in dialog systems are often no more than a few sentences in length.

We propose a novel approach that uses sentence embeddings to detect outliers in a corpus of short texts. We rank samples based on their distance from the mean embedding of the corpus and consider samples farthest from the mean outliers. Outliers come in two varieties: (1) *errors*, sentences that have been mislabeled whose inclusion in the dataset would be detrimental to model performance, and (2) *unique* samples, sentences that differ in structure or content from most in the data and whose inclusion would be helpful for model robustness. Building upon this approach, we propose a novel crowdsourcing pipeline that distinguishes errors from unique samples and uses the unique samples to guide workers to give more diverse samples.

Experimentally, we find that our outlier detection technique leads to efficient detection of both artificial and real errors in our datasets. We also use the proposed crowdsourcing pipeline to collect new datasets and build models for the dialog system tasks of intent classification and slot-filling. We find that the proposed pipeline produces more diverse data, which in turn results in models that are more robust.

2 Background and Related Work

2.1 Outlier Detection

Outlier detection (Rousseeuw and Leroy, 1987), also called outlier analysis (Aggarwal, 2015) or anomaly detection (Chandola et al., 2009), is the task of identifying examples in a dataset that differ substantially from the rest of the data.

For almost two decades, a body of work in NLP has investigated applying these ideas to data in order to identify annotation errors (Abney et al., 1999). Approaches have included the use of scores from trained models for POS tagging (Abney et al., 1999; Eskin, 2000; van Halteren, 2000; Dligach and Palmer, 2011; Fukumoto and Suzuki, 2004), count-based methods that compare examples from across the corpus (Nakagawa and Matsumoto, 2002; Hollenstein et al., 2016), characterizing data based on feature vectors projected down into a low-dimensional space (Guthrie et al., 2008), and tracking the difficulty of learning each example during training (Amiri et al., 2018). One particularly effective approach has been to find *n*-grams that match but have different labels, as shown for annotations including POS tags (Dickinson and Meurers, 2003), syntactic parses (Dickinson and Meurers, 2005; Dickinson, 2010; Dickinson and Smith, 2011), and predicate-argument relations (Dickinson and Lee, 2008). Our work instead uses continuous representations of text derived from neural networks.

While finding errors is an extremely useful application of outlier detection, also of interest are examples that are correct even though they are outliers, as these can be the most interesting and informative examples in a dataset. We term these examples *unique*. The problems of detecting and leveraging the unique examples in a dataset has received less attention, and the work that does exist focuses on identifying complete documents or segments of documents that are outliers out of a broader set of documents: Guthrie et al. (2007) used manually defined feature vectors to identify segments of documents with anomalous style, topic, or tone, and Kumaraswamy et al. (2015) and Zhuang et al. (2017) construct statistical models, identifying complete documents that are outliers within a set based on semantic variation.

Finally, a related but distinct topic is novelty detection (Soboroff and Harman, 2005; Lee, 2015; Ghosal et al., 2018), in which two sets of documents are provided, one that is assumed to be known, and one that may contain new content. The task is to identify novel content in the second set. While outlier detection methods are often applied to this problem, the inclusion of the known document set makes the task fundamentally different from the problem we consider in this work.

2.2 Data Collection

We build on prior work employing online crowd workers to create data by paraphrasing. In particular, we refine the idea of iteratively asking for paraphrases, where each round prompts workers with sentences from the previous round, leading to more diverse data (Negri et al., 2012; Jiang et al., 2017; Kang et al., 2018). We also apply the idea of a multi-stage process, in which a second set of workers check paraphrases to ensure they are correct (Buzek et al., 2010; Burrows et al., 2013; Coucke et al., 2018). Most notably, by incorporating our outlier detection method, we are able to automate detecting detrimental data points while also prompting workers in subsequent rounds to paraphrase more unique examples.

3 Outlier Detection

We propose a new outlier detection approach using continuous representations of sentences. Using that approach, we explored two applications: (1) identifying errors in crowdsourced data, and (2) guiding data collection in an iterative pipeline.

3.1 Method

We detect outliers in a dataset as follows:

1. Generate a vector representation of each instance.
2. Average vectors to get a mean representation.
3. Calculate the distance of each instance from the mean.
4. Rank by distance in ascending order.
5. (Cut off the list, keeping only the top k% as outliers.)

The final step is parenthesized as in practice we use a dynamic threshold approach, allowing the user to go through as much or as little of the list as they like.

The intuition behind this approach is that we expect our representations to capture the semantic structure of the space for each class. An example that is far away from other examples in the set is therefore less semantically similar in some sense, making it an outlier. Importantly, it may be an outlier for two distinct reasons: (1) it is not a valid instance of this class (i.e., an *error*), or (2) it is an unusual example of the class (i.e., *unique*).

This approach is applied independently to each class of data. As example applications we consider two dialog system tasks: intent classification and slot-filling. For classification, data for each possible intent label is considered separately, meaning we find outliers in the data by considering one intent class at a time. For slot-filling, we group the data into classes based on combinations of slots.

This outlier detection method is rather simple as it relies only on a sentence embedding method, a distance metric, and a threshold k; no other hyperparameters are involved. Moreover, the method requires no training. We shall see in Section 4 that this method performs well compared to baseline methods, no matter what sentence embedding method is used. We use Euclidean distance as our distance metric.[1]

3.1.1 Sentence Representations

Vector representation of sentences is an active area of research and we leverage the following approaches, each of which has been shown to have state of the art results in different use cases:

Universal Sentence Encoder (USE; Cer et al., 2018) A Deep Averaging Network method, which averages word embeddings and passes the result through a feedforward network. The USE is trained using a range of supervised and unsupervised tasks.

Smooth Inverse Frequency (SIF; Arora et al., 2017) A weighted average of word embeddings, with weights determined by word frequency within a corpus. We consider word embeddings from GloVe (Pennington et al., 2014) and ELMo (Peters et al., 2018).

Average An unweighted average of word embeddings. While simple, this approach has been shown to be effective for classification (Zhang and Wallace, 2017) and other downstream tasks (Zhu et al., 2018). Again, we consider GloVe and ELMo word embeddings as inputs.

3.1.2 Model Combination

In addition to ranked lists produced by using these core sentence embeddings, we also investigated aggregating the ranked lists using the Borda count, a rank aggregation technique that has previously been used for combining web search results (Dwork et al., 2001).

The Borda count aggregates multiple ranked lists of the same set of items into a single ranked list. First, points are assigned to each item in each list, with an item in position i in a ranked list of length N receiving $N - i$ points. Next, points for items are summed across all of the lists. Finally, the items are ranked by their total number of points, producing a final ranking.

3.2 Application: Error Detection

Our proposed use of outlier detection to identify errors requires no further processing. When used in practice, a user looks through the sorted list of examples, either stopping at a given fraction, or when errors become infrequent enough.

3.3 Application: Uniqueness-driven Data Collection

One core insight in this work is that outlier detection can be used for more than just finding errors. The outliers that are not errors are likely to be the most interesting and informative examples in our dataset. We propose to use these examples to guide data collection in an iterative process, with the goal of yielding more diverse data.

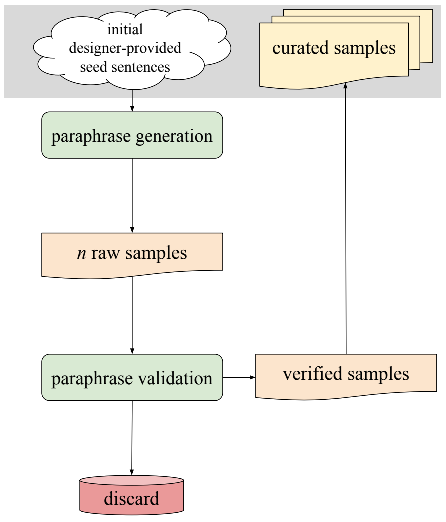
To demonstrate this idea, we developed a novel crowdsourcing pipeline for data collection. Following prior work in crowdsourcing for dialog (Kang et al., 2018; Jiang et al., 2017), we ask crowd workers to write paraphrases of seed sentences with known intents and slot values. This provides linguistic diversity in our data in a way that is easily explained to workers. For instance, given the seed sentence "*What is my savings account balance?*" a worker might write "*How much money do I have in my savings account?*".

Figure 1a shows a common crowdsourcing pipeline. The task designer writes seed sentences that target an intent (for classification) or a slot (for slot-filling). Crowd workers read the seed and write paraphrases. These paraphrases are then passed to another set of workers who validate if they are in fact accurate paraphrases.

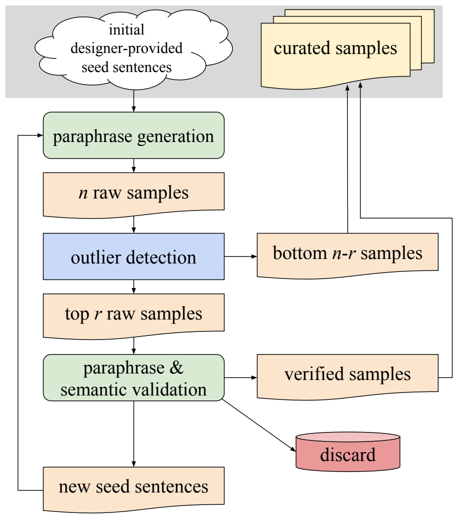
There are two major downsides to this standard pipeline. First, the validation step increases the cost-per-example. Second, the diversity of paraphrases depends on the given seed sentences (Jiang et al., 2017), creating a challenge for the task designer to think creatively.

We introduce a new pipeline, shown in Figure 1b that uses outlier detection to (1) reduce the number of sentences being checked, and (2) collect more diverse examples. Our new approach

[1] We found similar results in experiments with a density-based metric, Local Outlier Factor (Breunig et al., 2000).



(a) Standard Pipeline



(b) Our Uniqueness-driven Pipeline

Figure 1: Data collection pipelines. Our outlier detection method is incorporated into the uniqueness-driven data collection pipeline to guide crowd workers to write more diverse paraphrases. Boxes with rounded corners are manual processes performed by crowd workers, boxes with curved bases are data, and the box with square corners is our outlier detection method. In (b), r is the number of outliers detected from n samples.

uses outlier detection to select only a subset of sentences to be checked: namely, the ones ranked as most likely to be outliers. This reduces effort by focusing on the sentences most likely to be incorrect.

To try to increase diversity, we also introduce a process with several *rounds* of data collection. Outlier paraphrases collected in one round are used to seed the next round of data collection. We could directly use the sentences labeled as correct in the validation step, but while these sentences are correct, they may have diverged from the desired semantics (e.g. diverged from the desired intent class). To avoid confusion in the next round, we add a step in which workers are shown the most similar sentence from another intent (based on sentence embedding distance) and asked if the new seed is more similar to its intended intent or the alternative example. Only seeds judged as closer to their intended intent are retained.

This iterative process is intended to collect more diverse data by priming workers to think about ways of phrasing the intent that are not well covered in the current data. At the same time, we avoid the correctness issues Jiang et al. (2017) observed by incorporating the validation step.

4 Experiments

We perform two sets of experiments to probe the effectiveness of our outlier detection method. First, we consider error detection, comparing various ranking methods in artificial and real data scenarios. Second, we use our uniqueness-driven pipeline to collect data, measuring the impact on data diversity and model robustness. All experiments were conducted on English language data.

4.1 Error Detection

We measure error detection effectiveness in two settings, one artificial and the other more realistic.

Artificial First, following prior work, we consider an artificial dataset in which we inject noise by mixing data from different intents (Guthrie et al., 2007, 2008; Zhuang et al., 2017; Amiri et al., 2018). This provides an easy way to control the amount and type of anomalous data, but does lead to an easier task as the incorrect examples are generally more different than naturally collected errors would be. The specific data we consider is a set of 20 intents from an in-production dialog system. To generate outliers for a given intent class X_i, we randomly sample $p \cdot |X_i|$ sentences from other intents (e.g. $p = 0.04$, or 4%).

Real We collected a new set of sentences for ten intents. Workers were given three seed sentences per intent and asked to write five paraphrases per seed.[2] Each seed was given to 15 crowd workers, leading to 2250 samples overall, after which

[2] Crowd workers were paid 20¢ per HIT.

Method		MAP 1%	MAP 2%	MAP 4%	MAP 8%	MAP Real	Recall@k=10% 1%	Recall@k=10% 2%	Recall@k=10% 4%	Recall@k=10% 8%	Recall@k=10% Real
Baseline	Random	0.02	0.04	0.07	0.09	0.06	0.06	0.09	0.12	0.09	0.04
	Short	0.02	0.05	0.07	0.11	0.27	0.12	0.13	0.12	0.15	0.27
	Long	0.08	0.10	0.08	0.17	0.04	0.25	0.17	0.11	0.12	0.04
	BoW	0.14	0.14	0.14	0.24	0.10	0.41	0.24	0.24	0.22	0.10
Neural	Average GloVe	0.16	0.17	0.22	0.26	0.33	0.55	0.41	0.49	0.39	0.33
	Average ELMo	0.20	0.19	0.23	0.32	0.32	0.48	0.48	0.45	0.42	0.32
	SIF GloVe (SG)	0.35	0.29	0.41	0.45	0.21	0.75	0.64	0.62	0.54	0.20
	SIF ELMo (SE)	0.30	0.37	0.48	0.52	0.20	0.87	0.74	0.70	0.65	0.19
	USE (U)	0.46	0.50	0.63	0.69	**0.37**	0.84	0.83	0.80	0.74	**0.38**
Combined	U+SG	0.51	0.54	0.66	0.68	0.36	0.84	0.81	0.83	0.76	0.33
	U+SE	**0.58**	**0.62**	**0.68**	**0.72**	0.34	**0.89**	**0.87**	**0.86**	**0.81**	**0.38**
	U+SE+SG	0.51	0.57	0.67	0.69	0.36	0.87	0.84	0.84	0.78	0.34

Table 1: Outlier detection effectiveness for error detection in an artificial setting (constructed by randomly adding content from other intents) and a real setting (manually checked utterances from a crowdsourced set). The artificial results are represented by different values of p (1%, 2%, 4%, and 8%), where p represents different amounts of errors injected into each intent class. Our proposed neural methods are consistently more effective, reducing the manual effort required to identify errors.

Intent	Label	Example
withdrawal	*seed*	what is my withdrawal limit?
	inlier	how high is my withdrawal ceiling?
	error	How much money do I have available
balance	*seed*	what's my balance?
	inlier	Let me know how much money I have.
	error	What can I afford?
phone	*seed*	what's my bank's phone number
	inlier	I need to call my bank
	error	information on my bank
checks	*seed*	i need to order more checks
	inlier	I need to stock up on more checks
	error	No checkbox, more?

Table 2: Examples from the Real dataset. The "How much money do I have available" example was labeled an error since it is too similar to the balance intent. The "What can I afford?", "information on my bank", and "No checkbox, more?" examples are labeled as errors since they are too vague and ambiguous.

duplicates were discarded. To identify errors, the authors independently checked each sentence and discussed any disagreements to determine a consensus label (either *inlier* or *error*). Examples of seed sentences, along with inliers and errors is shown in Table 2.

4.1.1 Evaluation Metrics

Since our core outlier detection method produces a ranked list, we are interested in evaluating how effective it is at ranking errors near the top. We use Mean Average Precision (MAP) as an overall measure of list quality. MAP is the mean over intents of:

$$\frac{1}{|\text{errors for intent}|} \sum_{e \,\in\, \text{errors}} \frac{|\text{errors at or above } e|}{e}$$

where e is the position of an error in the list.

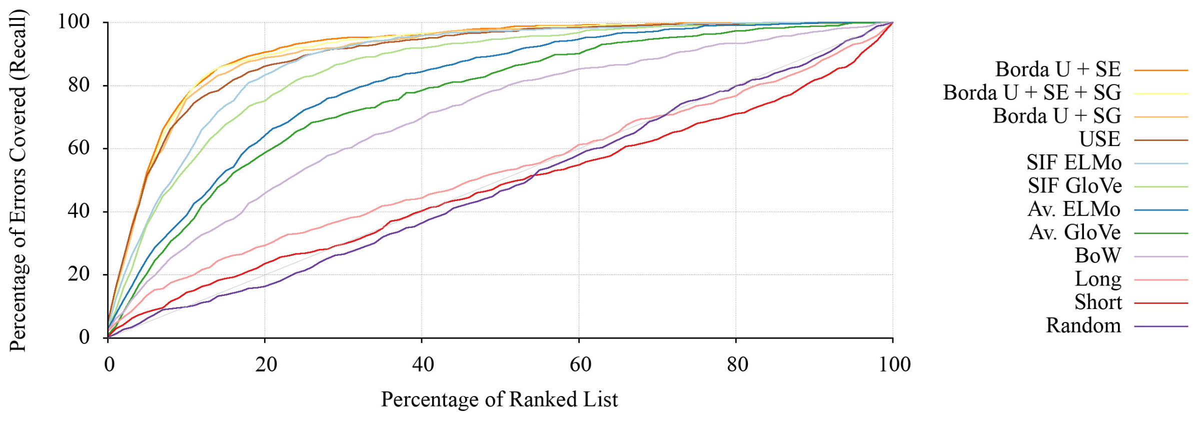
While this gives an overall qualitative measure for comparison, we are also interested in understanding the precision–recall tradeoff when choosing a threshold k on the ranked lists. We consider defining the cutoff as a percentage k of the list and measure the percentage of errors that are covered for each possible cutoff. This measure is equivalent to Recall@k, that is,

$$\text{Recall@}k = \frac{|\text{ errors above } k|}{|\text{errors}|}.$$

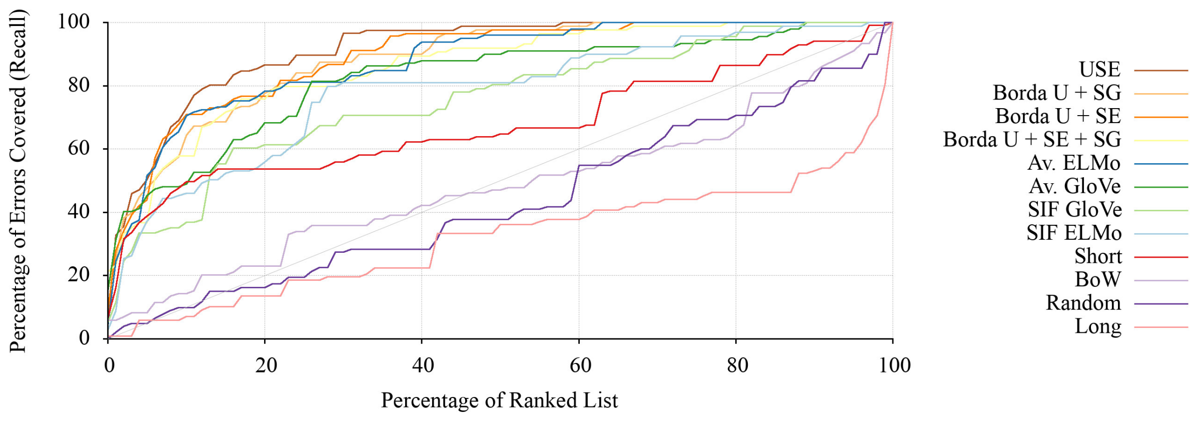
We average these values across intents to get an overall value for each cutoff percentage k.

4.1.2 Baselines

For comparison, we consider four simple baselines: randomly ordering the samples (Random), sorting from shortest to longest (Short), sorting from longest to shortest (Long), and calculating distances in the vector space defined by a bag of words (BoW).



(a) Artificial data.



(b) Real data.

Figure 2: Cumulative distribution of errors in ranked lists: a higher line indicates that the method places more errors earlier in the list. Results are less smooth for the real data as there are only 51 errors in the set. For each plot, the legend is in the same order as the lines at 20% (i.e., in (a) the top line is Borda U+SE, while in (b) it is USE). Neural ranking methods are consistently more effective, with USE covering over 80% of errors in the first 20% of the list.

4.1.3 Results

Table 1 presents MAP and Recall@k for error detection in the two settings (Artificial and Real). The neural methods outperform the baselines in both settings, demonstrating the effectiveness of our proposed approach. However, the relative performance of the neural methods differs substantially between the two settings. Specifically, (1) SIF performs better than an unweighted average on artificial data, but on real data we see the opposite trend, (2) combining rankings with Borda appears to help on the artificial data, but not on the real data, (3) ranking by length is surprisingly effective on the real data, and (4) results tend to be lower on the real data than the artificial (even at lower values of p). This last point suggests that the commonly used artificial setting does not perfectly capture the types of errors that occur in practice.

For (3), we note that the Short baseline method performs particularly well vis-à-vis other baselines on the real data, but not comparatively well on the artificial data. This can be explained by observing that the length of the average error in the real data is roughly 6 tokens, while the average inlier length is 8 tokens. Lengths of errors and inliers are roughly the same (roughly 8 tokens) in the artificial dataset, due to the outlier selection scheme.

While the values in Table 1 allow an overall comparison of the methods, they do not provide a clear qualitative sense of the distribution of errors in the lists. Figure 2 shows the distribution for each method in the two settings. The effective-

show average exchange rate from ten usd to cad last year

Figure 3: Example annotated sentence for the slot-filling task. The slot names are (in order of appearance) metric, amount, currency, and date.

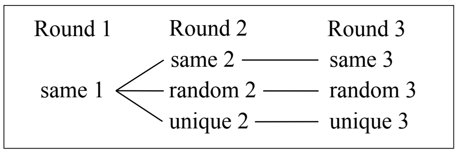


Figure 4: Data collection rounds. The final datasets combine data from all three rounds along each path.

ness of the neural methods, and USE in particular, is again clear. In the real data, when considering just the first 20% of the list, USE covers over 85% of the errors on average. One easy example was "*No checkbox, more?*" when the intent was to order more checks. This is clearly an error, which would at the very least need to have *checkbox* replaced by *checkbook*. In contrast, one hard example for USE was "*How much money do my banks*" when the intent was to request the user's balance. Until the last word, this example looks like it will be a valid balance request. These examples show that the system is qualitatively fitting our expectations for error detection.

4.2 Uniqueness-driven Data Collection

The second set of experiments evaluates our proposed uniqueness-driven data collection pipeline. We consider collecting data for two tasks used by dialog systems: intent classification and slot-filling. In each case, we calculate intrinsic measures of data diversity and the robustness of models trained on the data.

Tasks We consider intent classification with 10 intents related to banking, and slot-filling for foreign exchange rate requests with four slots: *amount*, *currency*, *date*, and *metric*. Figure 3 shows an example query with annotated slots.

Approaches As well as our proposed data collection pipeline (`unique`), we consider a variant where the next seed is chosen randomly (`random`), and one where the seeds are the same in every round (`same`). The third case is equivalent to the standard pipeline from Figure 1a. All three pipelines start from the same first round and then vary in the subsequent rounds, as depicted in Figure 4. Each pipeline collected data for three rounds. The final dataset for each approach combines data collected from all three rounds.

In both tasks, we asked workers to rephrase each seed sentence 5 times and showed each seed sentence to 15 workers. For classification there were 3 seed sentences per intent. For slot-filling

$$D(a,b) = 1 - \frac{1}{N}\sum_{n=1}^{N}\frac{|n\text{-grams}_a \cap n\text{-grams}_b|}{|n\text{-grams}_a \cup n\text{-grams}_b|}$$

$$\text{Diversity}(X) = \frac{1}{|I|}\sum_{i=1}^{|I|}\frac{1}{|X_i|^2}\left[\sum_{a\in X_i}\sum_{b\in X_i} D(a,b)\right]$$

$$\text{Coverage}(X,Y) = \frac{1}{|I|}\sum_{i=1}^{|I|}\frac{1}{|Y_i|}\sum_{b\in Y_i}\max_{a\in X_i}(1 - D(a,b))$$

Figure 5: Metrics for diversity and coverage from Kang et al. (2018). X and Y are sets of utterances labeled with intents from I, and X_i is the data in X for intent i. The distance metric for comparing a pair of utterances is based on the Jaccard Index over n-grams. We follow the work from Kang et al. (2018) and set N, the max n-gram length, to 3.

we defined 4 example scenarios, each corresponding to a specific combination of slots. We used Borda USE+SG with k set to 10% for the outlier detection model.

4.2.1 Evaluation Metrics

We consider several different metrics to probe how effectively our proposed pipeline improves data quality. In all cases, higher values are better.

Intrinsic We measure the diversity and coverage of each dataset using the metrics introduced in (Kang et al., 2018) and shown in Figure 5.

Extrinsic The main reason to increase dataset diversity is to construct more robust models. To directly evaluate that objective, we randomly divided the datasets collected by each pipeline into training and test sets (85-15 split). Our intuition is that a robust model should perform fairly well across all test sets. Training on a dataset that is not diverse will lead to a brittle model that only does well on data collected with the same seed sentences. For intent classification, we measure accuracy of two models: an SVM (Cortes and Vapnik, 1995) using bag of words feature representation, and FastText (Joulin et al., 2017), a neural network that averages across sentence embeddings and passes the result through feedforward layers. For slot-filling, we measure the F_1-score of a bi-directional LSTM with word vectors that are trained, but initialized with GloVe 300-dimensional embeddings. For all models, we average results across 10 runs.

	Data Collection Round			
	1	2	3	All
Diversity				
`same`	0.911	0.917	0.908	0.916
`random`	–	0.912	0.817	0.920
`unique`	–	**0.935**	**0.944**	**0.947**
Samples				
`same`	2040	2025	2024	5648
`random`	–	2010	2007	5747
`unique`	–	2083	1954	5922

Table 3: Classification: Diversity scores for data collected in each round (top), and the number of samples collected (bottom). The data for the All column combines the previous two sets in the row and the data from (`same` round 1). The `unique` approach produces data that is considerably higher diversity.

		Test Set		
Metric	Training	`same`	`random`	`unique`
SVM Accuracy	`same`	**0.99**	0.97	0.81
	`random`	0.98	**0.98**	0.81
	`unique`	0.99	0.97	**0.98**
FastText Accuracy	`same`	**0.98**	0.97	0.80
	`random`	0.98	**0.99**	0.83
	`unique`	0.98	0.98	**0.98**
Coverage	`same`	**0.68**	0.64	0.45
	`random`	0.67	**0.66**	0.44
	`unique`	0.64	0.58	**0.56**

Table 4: Classifier accuracy when training on one dataset and testing on another (top and middle), and coverage of the test set for each training set (bottom). As expected, the highest scores are when we train and test on the same data, but off the diagonal the `unique` test set (gray column) is considerably harder for models trained on other data while a model trained on `unique` performs consistently well. This accuracy trend is matched in coverage.

	Data Collection Round			
	1	2	3	All
Diversity				
`same`	0.916	0.911	0.893	0.909
`random`	–	0.913	0.910	0.915
`unique`	–	**0.926**	**0.935**	**0.930**
Samples				
`same`	994	911	952	2717
`random`	–	941	923	2808
`unique`	–	977	988	2914

Table 5: Slot-filling: Diversity scores for data collected in each round (top), and the number of samples collected (bottom). The data for the All column combines the previous two sets in the row and the data from (`same` Round 1). As seen for intent classification, the `unique` approach produces data that is of considerably higher diversity.

		Test Set		
Metric	Training	`same`	`random`	`unique`
Slot F_1	`same`	96.4	96.0	93.1
	`random`	96.4	**96.8**	93.6
	`unique`	**96.7**	96.5	**94.9**
Coverage	`same`	**0.812**	0.788	0.726
	`random`	0.736	**0.764**	0.660
	`unique`	0.761	0.752	**0.774**

Table 6: F_1-scores and coverage scores for each train-test pair for the slot-filling experiment. Training on the `unique` data produces a more robust model, with consistently high performance across test sets.

4.2.2 Results

Classification Table 3 presents the number of examples and diversity of data collected in each round with each approach. Diversity is consistently higher with seeds chosen using our proposed `unique` approach. Dataset sizes vary because of the removal of duplicates. The `unique` approach produces a larger final set as there is less duplication across rounds.

Table 4 displays accuracy scores and coverage for each combination of train and test sets. As expected, the highest scores are on the diagonal—training and testing on the same source data. More importantly however, training on the `unique` data produces a model that is robust, performing well across all three test sets. In contrast, training on the `same` or `random` data produces models that perform substantially worse on the `unique` test set. This trend is also present in the coverage scores in the bottom section of the table.

Table 7 shows some of the seed sentences produced by the `unique` and `random` approaches. These examples illustrate the trends in our metrics, with the seeds for the `random` approach often being very similar. Meanwhile, the `unique` approach produces seeds with grammatical variation and the introduction of quite different expressions, such as “ABA” instead of “routing number”.

Intent	Pipeline	Round 2	Round 3
routing	`random`	Where can I find the routing number for my bank? Do you have my bank's routing number? show me my banks routing number	what is the best routing number for my bank can you help find my bank routing number? I need to see the routing number for my bank
	`unique`	acquire my banks routing number how does a person find their correct routing number? I'm looking for the router number for my bank.	what is the correct ABA? How do I find the ABA? Whats the router number?
hours	`random`	Can you tell me when my bank is open? When does my bank open? What time is the bank open	How long is my bank open? What are the hours for my bank? What time will the bank be open
	`unique`	display when the bank closes What is the earliest you are open? look up the hours of operation for my bank	What hours do you carry what are your operating hours? What is the latest I can come in to a physical branch?
checks	`random`	i require ordering more checks. Get me more checks. can you explain to me how to order additional checks	I'd like additional checks Can you get me more checks? Please order more checks for me.
	`unique`	I need to stock up on more checks in what manner would i get more checks Could you rush me some more checks? I'm nearly out.	what is check ordering procedure? what is the fastest method to order checks? Teach me how to get more checks.

Table 7: Seed sentences for selected intents for the classification task. The `unique` approach leads to changes like the use of *ABA* for *routing* (top), and grammatical variations in the sentences for requesting checks (bottom). Examples of seeds for Round 1 include "what is my bank's routing number?", "when does the bank close?", and "how do i order more checks?".

Slot-filling Table 5 shows the number of samples collected per round for each of the data collection pipelines and the diversity of the sets. As in the classifier experiment, we observe that data produced by the `unique` pipeline is of higher diversity than the other two pipelines.

Table 6 displays F_1-scores and coverage for each train–test combination. Again, we see the same trends, with training on `same` or `random` leading to low results on the `unique` dataset, but not the reverse, and similarly for coverage, though the gaps are smaller than for classification.

5 Conclusion

Outliers are often the most interesting parts of our data, but outlier detection has received relatively little attention in NLP beyond its application to finding annotation errors. This paper introduces the first neural outlier detection method for short text and demonstrates its effectiveness across multiple metrics in multiple experiments.

We also propose a way to integrate outlier detection into data collection, developing and evaluating a novel crowdsourcing pipeline. This pipeline supports the creation of higher quality datasets to yield higher quality models by both reducing the number of errors and increasing the diversity of collected data. While the experiments discussed herein are concerned with components of dialog systems, we believe that similar data collection strategies could yield benefits to other areas of NLP as well.

Acknowledgments

The authors thank Yiping Kang, Yunqi Zhang, Joseph Peper, and the anonymous reviewers for their helpful comments and feedback.

References

Steven Abney, Robert E. Schapire, and Yoram Singer. 1999. Boosting applied to tagging and pp attachment. In *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Charu C. Aggarwal. 2015. *Outlier Analysis*, pages 237–263. Springer International Publishing, Cham.

Hadi Amiri, Timothy Miller, and Guergana Savova. 2018. Spotting spurious data with neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations (ICLR)*.

Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104.

Steven Burrows, Martin Potthast, and Benno Stein. 2013. Paraphrase acquisition via crowdsourcing and machine learning. *ACM Trans. Intell. Syst. Technol.*, 4(3):43:1–43:21.

Olivia Buzek, Philip Resnik, and Ben Bederson. 2010. Error driven paraphrase annotation using mechanical turk. In *Proceedings of the NAACL HLT Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for english. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP)*.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):15:1–15:58.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *Computing Research Repository (CoRR)*, abs/1805.10190.

Markus Dickinson. 2010. Detecting errors in automatically-parsed dependency relations. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Markus Dickinson and Chong Min Lee. 2008. Detecting errors in semantic annotation. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC)*.

Markus Dickinson and W. Detmar Meurers. 2003. Detecting errors in part-of-speech annotation. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Markus Dickinson and W. Detmar Meurers. 2005. Detecting errors in discontinuous structural annotation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Markus Dickinson and Amber Smith. 2011. Detecting dependency parse errors with minimal resources. In *Proceedings of the 12th International Conference on Parsing Technologies (IWPT)*.

Dmitriy Dligach and Martha Palmer. 2011. Reducing the need for double annotation. In *Proceedings of the 5th Linguistic Annotation Workshop (LAW)*.

Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. 2001. Rank aggregation methods for the web. In *Proceedings of the 10th International Conference on World Wide Web (WWW)*.

Eleazar Eskin. 2000. Detecting errors within a corpus using anomaly detection. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Fumiyo Fukumoto and Yoshimi Suzuki. 2004. Correcting category errors in text classification. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*.

Tirthankar Ghosal, Amitra Salam, Swati Tiwary, Asif Ekbal, and Pushpak Bhattacharyya. 2018. TAP-DLND 1.0 : A Corpus for Document Level Novelty Detection. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC)*.

David Guthrie, Louise Guthrie, Ben Allison, and Yorick Wilks. 2007. Unsupervised anomaly detection. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI)*.

David Guthrie, Louise Guthrie, and Yorick Wilks. 2008. An unsupervised probabilistic approach for the detection of outliers in corpora. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC)*.

Hans van Halteren. 2000. The detection of inconsistency in manually tagged text. In *Proceedings of the COLING Workshop on Linguistically Interpreted Corpora*.

Nora Hollenstein, Nathan Schneider, and Bonnie Webber. 2016. Inconsistency detection in semantic annotation. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC)*.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Youxuan Jiang, Jonathan K. Kummerfeld, and Walter S. Lasecki. 2017. Understanding task design trade-offs in crowdsourced paraphrase collection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Yiping Kang, Yunqi Zhang, Jonathan K. Kummerfeld, Parker Hill, Johann Hauswald, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2018. Data collection for dialogue system: A startup perspective.

In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of the 10th Machine Translation Summit*.

Raksha Kumaraswamy, Anurag Wazalwar, Tushar Khot, Jude Shavlik, and Sriraam Natarajan. 2015. Anomaly detection in text: The value of domain knowledge. In *Florida Artificial Intelligence Research Society Conference (FLAIRS)*.

Sungjin Lee. 2015. Online sentence novelty scoring for topical document streams. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Tetsuji Nakagawa and Yuji Matsumoto. 2002. Detecting errors in corpora using support vector machines. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING)*.

Matteo Negri, Yashar Mehdad, Alessandro Marchetti, Danilo Giampiccolo, and Luisa Bentivogli. 2012. Chinese whispers: Cooperative paraphrase acquisition. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Ines Rehbein and Josef Ruppenhofer. 2017. Detecting annotation noise in automatically labelled data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

P. J. Rousseeuw and A. M. Leroy. 1987. *Robust Regression and Outlier Detection*. John Wiley & Sons, Inc., New York, NY, USA.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ian Soboroff and Donna Harman. 2005. Novelty detection: The trec experience. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ye Zhang and Byron C. Wallace. 2017. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In *Proceedings of the 8th International Joint Conference on Natural Language Processing (IJCNLP)*.

Xunjie Zhu, Tingfeng Li, and Gerard de Melo. 2018. Exploring semantic properties of sentence embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Honglei Zhuang, Chi Wang, Fangbo Tao, Lance Kaplan, and Jiawei Han. 2017. Identifying semantically deviating outlier documents. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.