Automatic Quality Control of Transportation Reports using Statistical Language Processing

Matthew S. Gerber and Lu Tang

***Abstract*—The processes of developing, monitoring, and maintaining transportation systems produce large volumes of information. Human fieldworkers are often responsible for gathering this information and, despite their best efforts, will inevitably introduce errors into the collected data. This is a critical problem, since (1) the collected data are used to justify key infrastructure maintenance and development decisions, and (2) the volume of unstructured information (e.g., plain text) makes manual quality control prohibitively expensive. We introduce a solution to this problem in the example domain of vehicle accident reports. First, we analyzed a sample of accident reports and confirmed the existence of many data entry errors. Second, we developed and evaluated a statistical language processing approach that automatically identifies reports containing data entry errors. We tested a variety of system configurations on real-world data and compared their performance with multiple baseline methods. The best configuration achieved a performance score of 84%, far outperforming the baseline methods. Our results and analyses have quality control implications for any data source that pairs structured (e.g., coded fields) with unstructured text.**

***Index Terms*—Transportation reports, quality control, natural language processing**

I. Introduction

FIELDWORKERS routinely collect structured and unstructured data describing transportation infrastructure (e.g., a highway segment's physical condition) and associated events (e.g., vehicular accidents on a highway). Examples of structured data include bridge inspection dates and predefined traffic accident types, which are easily represented by a database schema that supports consistency checking and querying. Examples of unstructured data include plain-text descriptions written by fieldworkers as they observe the infrastructure and associated events (e.g., vehicular accidents). Analysts use both data types to schedule maintenance, perform safety analyses, and plan future infrastructure development projects.

These two data types – structured and unstructured – provide important, complementary views of transportation infrastructure and associated events; however, it is difficult to codify the meaning of unstructured data using a database schema. For example, assume that traffic accidents are described using structured fields (e.g., accident type) and unstructured English descriptions written by the reporting officer. Without additional processing, it is often impossible for the analyst to use these descriptions on a large scale, since he or she must read each description individually. This situation arises anywhere that structured and unstructured data coexist. Additional examples include (1) medical reports that document patient conditions and physician actions using a combination of lab values (structured) and narrative notes (unstructured), and (2) corporate reports that document revenue (structured) and managers' written annual assessments (unstructured). In all cases, the informal nature of unstructured data limits the data analyst's capabilities.

This article presents research that connects structured and unstructured transportation data for quality control purposes. Specifically, we have investigated automatic quality control for reports filed by police officers in response to vehicular accidents. Quality control is important for this dataset since it is used to plan safety measures and other infrastructure enhancements [1]. Each report characterizes an accident in roughly two ways: (1) with a combination of structured, fixed-choice data items (e.g., accident type), and (2) with an unstructured, written description of the accident that is supplied by the reporting officer. Example 1 shows an actual accident description:

(1) Vehicle 1 was traveling west on route 612, ran off the road to the left and struck a mobile home.

The reporting officer coded this accident as *roadway departure – right*. Such reports are internally inconsistent, since the written (unstructured) description disagrees with the codified (structured) description. Coding errors like the one in Example 1 can produce misleading conclusions that affect future decision making (e.g., by adding uncertainty to the assessment of left- versus right-sided hazards). Thus, it is critical to identify inconsistent reports, either for correction or simply to exclude them from the decision making process. As shown in Example 1, it can be straightforward for a human analyst to inspect the structured and unstructured portions of a report for consistency; however, given the amount of data collected, it is impossible to manually inspect all reports, and smaller subsets can only be checked at great expense.

In response to the problems described above, we have developed an approach that uses methods from natural language processing and data mining to automatically identify internally inconsistent traffic accident reports (i.e., those whose structured and unstructured views disagree). We have evaluated our approach on a sample of reports collected in 2010 by the Virginia Department of Transportation (VDOT). Our experiments show that simple linguistic features can be used effectively in a statistical prediction model: we achieved a performance score of 84% on the task of identifying inconsistent

M. Gerber is with the Department of Systems and Information Engineering, University of Virginia, USA (e-mail: msg8u@virginia.edu).

L. Tang is with the Department of Statistics, University of Virginia, USA (e-mail: lt7pv@virginia.edu).

reports, outperforming two baseline methods. Our approach is not specific to the domain of traffic reports; it can be applied to any problem where transportation infrastructure and events are monitored using a combination of structured, non-linguistic data and unstructured, linguistic data. Identifying inconsistencies in these data will improve the monitoring process and strengthen subsequent analyses. We are not aware of previous investigations targeting this important task.

This article is organized as follows. First, in Section II, we review related work in other domains. In Section III, we describe the experimental data that we used in our study. In Section IV, we describe our approach, which we evaluate in Section V. We discuss the outcomes of our experiments in greater detail in Section VI and conclude in Section VII.

II. Related Work

Previously, most research in quality control has proceeded independently of natural language processing. However, the widespread adoption of digital report formats offers a unique opportunity to combine these fields for the purpose of automated, scalable quality control. Below, we briefly review important work in the two domains and then discuss the small amount of existing research that combines them. We discuss points of departure and novelty of the current research throughout.

A. Quality Control

Quality control techniques are routinely used to ensure electronic data quality. This is a crucial task owing to the rise in quantity and importance of such data. Batini et al. [2] provide an extensive discussion of data quality assessment and improvement methodologies. Most of these strategies are data-driven and use data source trustworthiness, data standardization, data fusion, and cost management techniques [3]. Process-driven strategies, on the other hand, aim to redesign the process of data creation or modification [2]. In the present research, we have designed a monitoring approach that improves the quality of an existing dataset by automatically identifying data inconsistencies. Our approach is data-driven; however, it is based on a novel application of natural language processing techniques.

As described by Even and Shankaranarayanan [4], the cost-quality trade-off is a key consideration when evaluating alternative quality control measures. Effective measures will be underutilized if such costs are prohibitive. Thus, the present research focuses on producing quality control measures that incur minimal long-term costs. We base our approach on statistical learning methods that, once deployed, can automatically identify data inconsistencies with no human intervention. This approach is not specific to our study domain and can be extended to other domains with similar data environments (see Section III for a description of the data). These features of our approach contrast with many other quality control procedures (e.g., report inspection) that require extensive human labor and are not portable across domains [5].

B. Natural Language Processing

The present research relies heavily on natural language processing (NLP) techniques that infer structure from unstructured expressions of natural language (e.g., human-written reports). The field of NLP has changed dramatically in the past decade, focusing on robust statistical techniques instead of the brittle, handwritten processing rules developed previously (see Jurafsky and Martin [6] for a detailed presentation of NLP). The statistical NLP approach has found widespread adoption in addressing a variety of textual problems such as search [7], clustering [8], and information extraction [9]. Underlying statistical techniques like naive Bayesian classification and support vector machines often produce better NLP output than handwritten rules and are still amenable to formal evaluation metrics like accuracy, F-measure, and information gain [10]. Statistical NLP is now sufficiently mature for application to problems like automatic quality control of textual documents – the focus of the present research.

C. NLP for Quality Control

As noted above, quality control research and NLP research have proceeded, for the most part, separately to date. One exception is the work of Jeske and Liu [11], who used NLP techniques to detect failure cases within an aircraft inspection database. The reports were collected from inspection logs and comprised coded information as well as written descriptions of an airplane's condition. Jeske and Liu applied text classification techniques to sort these reports into predefined categories. In their approach, word occurrences were extracted from the unstructured text and used as input to a naive Bayesian classifier trained on a hand-labeled portion of the corpus. The authors measured classification performance by accuracy, finding that failure reports could be identified with 91.8% accuracy. Their accuracy value is not comparable with ours since the datasets and metrics are different.

Although our approach (see Section IV) is similar to that of Jeske and Liu, our objective is much different. We are not concerned with identifying a particular class of report, for example speeding tickets. Rather, we are interested in detecting internal inconsistencies across a range of report types. We believe this is a more difficult task but also one that can be addressed effectively using statistical NLP methods. Furthermore, Jeske and Liu used simple linguistic features to represent textual information. We have investigated word sequence information (versus isolated word occurrences), synonymy information, and dimensionality reduction techniques, all of which are new within automated quality control. Our evaluation results show that these additional information sources and techniques significantly improve the system's ability to recognize inconsistent transportation reports.

III. Experimental Data

Over the last 20 years, the Virginia Department of Transportation (VDOT) has digitized more than 2 million crash reports, each of which documents a single accident involving one or more vehicles. These reports capture important information that is used for safety planning and development purposes.

For example, the following report excerpt shows the officer's written narrative and three fixed-choice selections:

(2) *Vehicle lost control on the ice went off the road and over turned.*

- Driver's Action: Fail to Maintain Proper Control
- Vehicle Maneuver: Going Straight Ahead
- Crash Type: Head On

In Example 2, the fixed-choice coding deviates from what is written in the narrative: it is unlikely that a head-on collision between two vehicles occurred, given the written description.

There are a few possible conclusions with regard to such discrepancies. First, the officer might have misunderstood the intended meaning of the fixed-choice item and selected incorrectly. Second, the officer might have written an imprecise narrative that does not appear to agree with the fixed-choice characterization, this despite the narrative's actual fidelity. Third, the officer might have simply written an inaccurate description (e.g., by misremembering the accident). We believe each of these discrepancies should be of concern to the database maintainers and thus do not differentiate between them. In the past, VDOT personnel have manually checked for these discrepancies within small subsets of the database; however, it is not feasible to extend this costly process to cover all reports. Our approach performs the same task automatically.

A. Data Sampling

We obtained a sample of the VDOT crash report database containing 115,990 reports covering the 2010 calendar year (approximately 318 reports per day). To simplify our analyses, we filtered out reports for multi-vehicle crashes as well as those containing corrupted/unreadable text.[1] This left us with 31,317 readable, single-vehicle reports (27% of the original collection). From this subset, we randomly selected 1,424 reports for model building and evaluation. Each report in this dataset contains a *CRASH TYPE* coding selected by the police officer and a narrative description of the accident. Table I shows the coding distribution observed within our dataset. We selected *CRASH TYPE* because this field can often be inferred from written narratives, which have an average length of 27 words. In cases where a crash involved multiple events (e.g., a rear-end collision followed by a roadway departure), the type of the first harmful event is recorded as the *CRASH TYPE*. We also used this rule in our annotation effort, which is described below.

B. Annotation

We based our approach on supervised statistical NLP and thus had to develop a human-annotated corpus of crash reports for use in training and evaluating our models. Two annotators were asked to read the written narratives in the 1,424-report sample and determine the best *CRASH TYPE* values according to the written narrative descriptions. We assessed inter-annotator agreement using Cohen's kappa coefficient [12], producing a value of 0.97 (high agreement). This test indicates that the *CRASH TYPE* field can be reliably and independently inferred from the reporting officer's written description. We then compared the annotated *CRASH TYPE* values with those selected by reporting officers and found that only 83.8% of the reports in our sample dataset were internally consistent. Table II shows the comparison of police officers' coding and the annotated values derived from the associated narratives. The diagonal cells represent agreements whereas the off-diagonal cells represent inconsistencies. Approximately 16% of the reports in our sample exhibited internal inconsistencies such as the ones in Examples 1 and 2. This level of inconsistency motivates the development of a model for automatic inconsistency detection, which we describe in the following section.

[1]We removed multi-vehicle crashes because the complexity of their associated reports could have confounded some of the more basic insights we sought in this preliminary work. We leave multi-vehicle reports for future work.

IV. Methods

We used the data described in Section III to build a statistical model for identifying inconsistent reports. In Section IV-A, we present the model in general terms. In the sections that follow (IV-B and IV-C) we describe the model's input and construction. Section V presents the evaluation metrics and results for various model configurations. We focused exclusively on the *CRASH TYPE* field in our research – other fields and report types (i.e., those from other domains) can be treated similarly.

A. A Model of Inconsistency

An accident report is consistent if the coded *CRASH TYPE* field agrees with the written narrative. Thus, to automatically detect inconsistent reports, it is sufficient to classify narratives into the crash types shown in Table I and compare the output with the police coding. Our statistical model for narrative classification takes the following form:

$$\widehat{CRASH\ TYPE} = f(PoliceCoding, \\ LinguisticFeatures) \quad (3)$$

Equation 3 has three components. On the left-hand side is a categorical response variable indicating the narrative's predicted type. This prediction is a function f of the reporting police officer's coding as well as linguistic features extracted from the narrative text. We include the police coding as an explanatory variable for two reasons. First, the police coding is correct 83.8% of the time and the model should take this into account. Second, there are patterns in the 16.2% of reports that are inconsistent. For example, an officer might routinely mis-code an accident of a particular type. Our model is able to identify these patterns and use the police coding to infer the correct accident type. We describe the linguistic features in the following section.

B. Linguistic Feature Extraction

Statistical models typically operate over numeric objects. For example, we might model the height of individuals in a population in order to make claims about the likelihood of observing a particular height. Written narratives, however, are not inherently numeric, since they only contain words and

TABLE I
CRASH TYPE CODING OPTIONS AND SAMPLE DISTRIBUTION

Option	Crash Type	Count
1	Rear End	10
2	Angle	42
3	Head On	49
4	Sideswipe - Same Direction	7
5	Sideswipe - Opposite Direction	2
6	Fixed Object - In Road	25
7	Train	0
8	Non-Collision	82
9	Fixed Object - Off Road	890
10	Deer	208
11	Other Animal	14
12	Pedestrian	49
13	Bicyclist	0
14	Motorcyclist	1
15	Backed Into	1
16	Other	44
Total		**1424**

TABLE II
ORIGINAL POLICE CODING ERROR DISTRIBUTION

		Human-coded Narratives															
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Police Officers' Coding	**1**	0	0	0	0	0	1	0	0	8	0	0	0	0	0	1	0
	2	0	0	0	0	0	1	0	3	34	1	0	1	0	0	2	0
	3	0	0	0	0	0	2	0	2	35	7	2	1	0	0	0	0
	4	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0
	5	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0
	6	0	0	0	0	0	12	0	1	11	0	0	0	0	0	1	0
	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	8	0	0	0	0	0	1	0	67	14	0	0	0	0	0	0	0
	9	0	0	0	0	0	1	0	35	845	2	2	0	0	0	5	0
	10	0	0	0	0	0	0	0	0	2	206	0	0	0	0	0	0
	11	0	0	0	0	0	0	0	1	0	0	13	0	0	0	0	0
	12	0	0	0	0	0	0	0	0	0	0	0	49	0	0	0	0
	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	16	0	0	0	0	0	6	0	19	18	0	0	1	0	0	0	0

various characters. To create the model shown in Equation 3, we must therefore transform each narrative into a numeric representation (i.e., $LinguisticFeatures$) – this is the process of feature extraction. In the simplest case, one would represent each narrative as a vector in n dimensions, with n being the number of unique words in the narrative collection. The value along each dimension is set to the frequency of the associated word in the narrative. The model assigns a coefficient to each dimension in order to estimate the probability of each *CRASH TYPE*. For example, a narrative with large counts for "ran" and "off" might belong to *CRASH TYPE* 9 (Fixed Object - Off Road). One can extract more complicated features; however, the general idea remains the same. In the following sections, we discuss the features extracted from each narrative.

1) Preprocessing: According to the grammatical rules of English (or any language), a word can exist in multiple forms. For example, the verb *strike* can appear as *strike*, *strikes*, *struck*, and *striking* depending on the context in which it appears. Intrinsically, however, these words mean approximately the same thing and so should be treated as the same word. We achieved this with word stemming, a process that transforms different forms of words into their plain forms as shown in the following examples:

(4) Original text: *V1 hit black ice ran off road right and struck the guardrail then overturned.*

(5) Stemmed text: *V1 hit black ice run off road right and strike the guardrail then overturn.*

In addition to stemming, we removed very common words (e.g., "the" and "is") known as stopwords. Each of these preprocessing steps eliminates unnecessary variation from the narratives, making the estimation of Equation 3 more tractable.

2) Features:

a) Unigrams: As a starting point, we use unigram features similarly to Jeske and Liu [11]. These features capture the presence of single words in isolation, without regard for the order of the words. These features are easy to extract; however, certain aspects of meaning are ignored, for example the distinction between passive and active voice as shown in the following examples:

(6) *Vehicle hit deer.*

(7) *Deer was hit by the vehicle.*

These two narratives have the following 3-dimensional representations (following stemming and stopword removal):

Example	deer	hit	vehicle
6	1	1	1
7	1	1	1

As shown, the differences in word order are not reflected in the vector-based representations, making the two descriptions identical (a desirable property, in this case).

b) Bigrams: In Examples 6 and 7, the unigram representation correctly identifies the similarity in meaning between the two narratives; however, this is not always the case, since word order often plays a crucial role in meaning. Consider the following examples:

(8) *The vehicle avoided a tree and struck a bicyclist.*
(9) *The vehicle avoided a bicyclist and struck a tree.*

Examples 8 and 9 clearly have different meanings that the unigram model would not recognize. To address this shortcoming, we considered word sequences of length two. The bigram vector representations for Examples 8 and 9 are shown in Table III. As shown, the vector representations for these examples are quite different. One can consider higher-order n-grams (e.g., trigrams or quadrigrams); however, these features become increasingly sparse and less informative, so we limited our approach to unigrams and bigrams only.

c) Synonym Sets: Many words mean roughly the same thing despite the fact that they do not share a common word stem. Consider the following narratives, which demonstrate this point:

(10) *Motorcycle hit guardrail.*
(11) *Car struck guardrail.*

If the car-motorcycle and hit-struck distinctions were inconsequential to the analyst, Examples 10 and 11 would indicate roughly the same type of event. This is due to synonymy relationships that hold between the subjects and verbs in the two sentences. The WordNet database [13] contains a comprehensive list of synonym relationships that organize words into synonym sets (or synsets). For example, the synset { *hit*, *strike*, *touch*, ... } includes the verbs from the examples above and is defined as “make physical contact with”. WordNet also defines hierarchical relationships between words, for example that motorcycles and cars are *types of* vehicle. Using WordNet synsets and hierarchical relationships, we can normalize Examples 10 and 11 as follows:

(12) {*Vehicle*} {*touch*} *guardrail.*
(13) {*Vehicle*} {*touch*} *guardrail.*

The motivation for this normalization is similar to that of stemming: by reducing unnecessary variation in the language used to describe accidents, our model (Equation 3) will be easier to estimate and apply. Lastly, note that we also investigated the use of bigrams following the synset transformation, resulting in bigrams such as “{*vehicle*}-{*touch*}” for the examples above.

C. Model Building

Given the annotated accident reports (Section III-B) and extracted features (Section IV-B), the model learns a coefficient for each feature indicating the associated crash type. A primary impediment to this task stems from the number of linguistic features extracted from the narratives. In our dataset, the 1,424 sample narratives produce a more than 20,000 features, since each unigram, bigram, synset, and synset bigram constitutes a feature. Thus, we have many more features than reports and estimating Equation 3 with confidence becomes infeasible. In order to train our model, we first reduced the number of features and retained the most informative ones (Section IV-C1). Following this, we trained the model using logistic regression (Section IV-C2).

1) Feature Selection: We experimented with three feature selection methods when building our narrative classification model. We describe these methods below.

a) Collinearity Filter: Many features in our dataset are collinear (i.e., redundant) across narratives. For example, any narrative with one occurrence of the bigram *run off* will also have a unigram feature *run* with value 1. Since the *run off* bigram is very common, many narratives will have the same value for these two features. We addressed this problem by filtering out collinear features in the following way. First, we constructed a matrix in which each column is a vector representing a language feature and each row is a narrative. The value in each cell is 1 if the feature is present in the narrative and 0 otherwise. Thus, each column is a binary vector in an n-dimensional space, where n is the number of narratives. We then estimated the collinearity of two columns (i.e., features) $\overrightarrow{f_1}$ and $\overrightarrow{f_2}$ using cosine similarity:

$$\begin{aligned} S(\overrightarrow{f_1}, \overrightarrow{f_2}) &= cos(\theta_{\overrightarrow{f_1}\overrightarrow{f_2}}) \\ &= \frac{\overrightarrow{f_1} \cdot \overrightarrow{f_2}}{\|\overrightarrow{f_1}\|\|\overrightarrow{f_2}\|} \end{aligned} \tag{14}$$

In Equation 14, $\theta_{\overrightarrow{f_1}\overrightarrow{f_2}}$ is the angle formed between the two feature columns, $\cdot$ is the dot-product, and $\|\overrightarrow{f_i}\|$ is the Euclidean length of a feature column. The function S ranges from 0 to 1 with the latter indicating maximal similarity (collinearity) between the features.

We computed Equation 14 for every pair of feature columns and grouped features with a similarity score above 0.97. We then randomly selected a single feature from each group and removed the remaining features from the group, reducing the number of features from 20,560 to 5,216.

b) Principal Component Analysis: Principal component analysis (PCA) is a method for identifying feature combinations that effectively describe the data. For example, instead of using two separate features – say, a bigram and unigram feature – we can combine these features into a single feature. This has the benefit of further reducing the number of features while retaining as much detail as possible in the feature-based representation. A full description of PCA is beyond the scope of this article, and we refer the interested reader to the relevant literature [14], [15]. We used the PCA package “FactoMineR” provided by the R statistical language to extract principal components from the vector-based representation demonstrated in Table III.

c) Stepwise Selection: In addition to the above methods, we experimented with forward stepwise feature selection following Pudil et al. [16]. Forward selection starts by building a separate model from each of the n features and selecting

TABLE III
BIGRAM NARRATIVE REPRESENTATION FOR EXAMPLES 8 AND 9.

Example	vehicle-avoid	avoid-tree	tree-strike	strike-bicycle	avoid-bicycle	bicycle-strike	strike-tree
8	1	1	1	1	0	0	0
9	1	0	0	0	1	1	1

the best model according to its performance on a test set. Next, each of the $n-1$ remaining features is added to the selected model, producing $n-1$ models that are evaluated. This process repeats until no performance gains are observed by adding additional features (or all features are selected). This approach can be computationally intensive with large feature sets such as ours; however, it directly maximizes the model's performance on a test set, which is a beneficial property.

2) Classification Models: As described in the following section, we experimented with various combinations of feature types (e.g., bigram versus unigram) and feature selection techniques. Regardless of the combination, we used the resulting data to train a multi-class logistic regression model of Equation 3 using the implementation provided by LIBLINEAR [17]. We also experimented with multinomial log-linear models [18] and support vector machines [19] provided by R; however, the large feature spaces caused performance issues for the latter models. Specifically, the running time using log-linear models and support vector machines is five times longer than that of using LIBLINEAR with no appreciable performance gain. We selected LIBLINEAR for all experiments since it is specifically designed for large, sparse datasets.

V. Evaluation and Results

A. Setup

Our evaluation setup requires a collection of testing accident reports and a classification model to process them. The testing reports will not have been seen by the model during the training phase, and the model's task is defined as follows:

> Given a testing accident report, determine whether the coded *CRASH TYPE* field agrees with the *CRASH TYPE* implied by the written narrative. If these two values disagree, flag the report as inconsistent.

After applying the model to the testing reports, each report is associated with three values:

1) **Police Label:** the coded *CRASH TYPE* value selected by the reporting officer
2) **True Label:** the human-annotated *CRASH TYPE* value derived from the narrative
3) **Predicted Label:** the *CRASH TYPE* predicted by the model

If values 1 and 3 match, the model predicts that the report is consistent; otherwise, the model flags the report as inconsistent. The model's prediction will be correct if and only if values for 1 and 2 indicate a similar consistency status. The following examples demonstrate correct and incorrect predictions, respectively:

(15) **Correctly predicting inconsistency**
Police Label: Fixed Object-Off Road
True Label: Deer
Predicted Label: Deer

(16) **Incorrectly predicting consistency**
Police Label: Fixed Object-Off Road
True Label: Fixed Object-In Road
Predicted Label: Fixed Object-Off Road

B. Baseline Models

We implemented two baseline models with which to compare our statistical model. The first baseline is a trivial model that identifies every report as inconsistent, regardless of the report's content. We obtained a stronger baseline model by creating rules for identifying inconsistent reports. This is possible since some police coding errors show an observable pattern. Returning to Table II, we observe that police codings of 1-5 are always incorrect. These codings deal with multi-vehicle accidents (e.g., head-on collisions), which do not exist in our evaluation dataset and represent errors on the part of the reporting officer. Thus, the second baseline flags as inconsistent all reports with police codings 1-5.

C. Metrics

For each report, the model outputs a value indicating consistency or inconsistency. Each output value is either correct or incorrect, creating four possible outcomes for each report. Table IV summarizes these outcomes. We used the F-measure score over the outcomes in Table IV to summarize a model's performance. First, we define precision and recall as follows:

$$Precision = \frac{TI}{TI + FI}$$
$$Recall = \frac{TI}{TI + FC}$$

In words, precision is the percentage of predicted inconsistent reports that are truly inconsistent, and recall is the percentage of truly inconsistent reports that are flagged by the model. F-measure combines these two scores into a single scalar value defined as follows:

$$F = 2 * \frac{Precision * Recall}{Precision + Recall}$$

The F-measure ranges from 0-1 (inclusive) and is only large when both precision and recall are large. To better understand these metrics, consider the trivial baseline that flags all reports as inconsistent. This model has perfect recall, since FC equals zero resulting in a recall of 1; however, precision is very low because FI is large for this model, resulting in a low F-measure.

We evaluated our models using 10-fold cross-validation over the human-annotated accident reports. We divided our human-annotated accident reports into 10 equally sized parts, or folds.

TABLE IV
CONTINGENCY TABLE OF OUTCOMES RESULTING FROM A MODEL PREDICTION, WHERE *T = True Label*, *M = Predicted Label*, AND *P = Police Label*.

Outcome	True versus Police	Predicted versus Police
True Inconsistent (TI)	$T \neq P$	$M \neq P$
True Consistent (TC)	$T = P$	$M = P$
False Inconsistent (FI)	$T = P$	$M \neq P$
False Consistent (FC)	$T \neq P$	$M = P$

We then used each fold as testing data for a model trained on the remaining 9 folds and averaged the precision, recall, and F-measure scores across all testing folds to arrive at a final score for the model. This approach has two key benefits. First, it allows one to evaluate a trained statistical model over all of the data. Second, it prevents the model from "cheating" by making predictions about reports it saw during training time.

D. Primary Results

Table V shows the cross-validation results for the baseline models as well as various configurations of the statistical model, which uses the LIBLINEAR classifier. The *Features* column indicates the set of features used within the statistical models. In the *Model* column, *Filter* denotes the cosine collinearity filter, *PCA* denotes principal component analysis, and *Stepwise Selection* denotes forward stepwise selection. During development, we evaluated the effectiveness of 30 to 100 principal components by increments of 5 and selected the number that produced the best results on a development dataset. The number of principal components we choose is different for each model and is indicated in parentheses in Table V.

We tested the significance of the differences between models in Table V using a bootstrapped percentile test [20], which uses resampled (with replacement) data to repeatedly measure the differences in F-measure between two models. We calculated the exact p-value for the hypothesis $H_0 : D_F(A, B) = 0$, where D_F denotes the true difference in F-measure between model A and model B. Our tests indicated that all 6 statistical models in Table V are significantly better than both baselines, with p-values smaller than 0.0001.

VI. DISCUSSION

The results in Table V demonstrate the power of language information for automatic quality control of transportation reports. All models (1-6) that use language features are significantly better than our baseline models that do not use language features, including model 2, which uses handwritten rules to identify inconsistent reports. To further understand the strengths and weaknesses of different language features and reduction techniques, we compared pairs of models. The results of these comparisons are shown in Table VI.

We observed that the stepwise selection models (2, 4, and 6) are significantly better than the principal component models (1, 3, and 5), regardless of the features and number of principal components used (all comparisons had p-values less than 0.0001). It is difficult to identify the underlying cause of this advantage, since the output of PCA (i.e., linear combinations of features) is not readily interpretable. In addition to better performance, the stepwise model does not combine features and the resulting model remains interpretable. The disadvantage of the stepwise model is that it requires substantial computational time to select a small subset of features from a very large pool of candidates (i.e., selecting a set of 25 features from 20,000 features took approximately 24 hours on a dual-core CPU with 32GB of RAM). Our collinearity filter helps with this by removing redundant features and reducing the runtime by more than 60% from 20,000 to 5,000 features and from 24 hours to about 8 hours.

The use of bigram information, however, does not improve modeling performance (compare models 2 and 4). In this case, unigrams alone are sufficient to interpret the internal consistency of reports. Synonymy information also does not improve modeling performance (compare models 4 and 6). We believe this is due to the low variation in word choice used by reporting officers, who, for example, uniformly use the word *vehicle* instead of alternatives such as *car*. Synonymy information contributes little information to the model in such cases. These two negative results are important since bigram information can produce very large feature spaces that are difficult to model, and synonym information can be difficult to extract. As shown in Table VI, neither of these efforts is beneficial.

We analyzed model 6 (the best in Table V) by inspecting the model coefficients assigned to the first 10 selected features. Table VII shows the results, where a more positive coefficient favors the listed class and a more negative coefficient disfavors the listed class. Most of the coefficients are negative, but it is the relative value (i.e., more or less negative) that matters when differentiating the classes. Although it is not possible to give an intuitive interpretation for all of the coefficients, some stand out. For example, a police coding of *Deer* heavily disfavors the types *FixedObjectOffRoad* and *Pedestrian* versus *Deer* and *OtherAnimal*. The unigram *strike* feature, on the other hand, is most favorable for *Pedestrian* and *FixedObjectInRoad* (e.g., "the vehicle struck the runner") and least favorable for *Motorcyclist*. These observations demonstrate that, in some cases, the model reflects our intuition about how variation in officers' language use affects the classification of narrative reports. Practically speaking, this implies that report writing guidelines could help improve the detection of inconsistent reports by suggesting controlled terminology for specific accident types.

As noted in Section III-A, we focused this initial research on single-vehicle crash reports. We suspect that single-vehicle reports are easier to analyze than multi-vehicle reports, and so our model might not perform as well when directly applied to the full set of reports; however, we hypothesize that similar results will be obtained by first retraining the model on an

TABLE V
COMPARISON OF MODELING RESULTS. *PCA(n)* REFERS TO PRINCIPAL COMPONENT ANALYSIS USING THE TOP n COMPONENTS.

Features	Model	Precision	Recall	F-measure (%)
N/A	Baseline 1 (flag all reports)	16.15	100	27.81
PoliceCoding	Baseline 2 (simple rule)	100	47.83	64.71
PoliceCoding+Unigrams	**1**. Filter + PCA(85)	77.88	76.52	77.19
PoliceCoding+Unigrams	**2**. Filter + Stepwise Selection	91.75	77.39	83.96
PoliceCoding+Unigrams+Bigrams	**3**. Filter + PCA(55)	83.18	77.39	80.18
PoliceCoding+Unigrams+Bigrams	**4**. Filter + Stepwise Selection	91.37	78.26	84.31
PoliceCoding+Unigrams+Bigrams+Synsets	**5**. Filter + PCA(65)	83.18	77.39	80.18
PoliceCoding+Unigrams+Bigrams+Synsets	**6**. Filter + Stepwise Selection	91.37	78.26	84.31

TABLE VI
COMPARISON OF MODELS AND RESULTING CONCLUSIONS. THE FIRST COLUMN INDICATES THE MODELS BEING COMPARED, THE SECOND COLUMN INDICATES WHICH OF THE TWO MODELS PERFORMED BEST ACCORDING TO A TEST OF STATISTICAL SIGNIFICANCE (P-VALUE LISTED IN PARENTHESES), AND THE THIRD COLUMN PROVIDES CONCLUSIONS DRAWN FROM THE TEST. ALL CONCLUSIONS ARE WITH RESPECT TO THE TASK OF AUTOMATIC INCONSISTENCY DETECTION (VERSUS NLP MORE GENERALLY).

Models	Best model (p-value)	Conclusions
1,2	2 ($p < 0.0001$)	Stepwise selection is better than PCA
3,4	4 ($p < 0.0001$)	
5,6	6 ($p < 0.0001$)	
2,4	Inconclusive ($p > 0.05$)	Bigrams do not improve stepwise selection models
4,6	Inconclusive ($p > 0.05$)	Synsets do not improve stepwise selection models

TABLE VII
COEFFICIENTS FOR THE FIRST 10 FEATURES SELECTED FOR THE BEST MODEL IN TABLE V (MODEL 6). POSITIVE/NEGATIVE COEFFICIENTS INDICATE HIGHER/LOWER LIKELIHOOD OF THE CLASS LISTED IN THE FIRST COLUMN.

Class	Feature Coefficients									
	Bigram	PoliceCoding					Synset		Unigram	
	tree-fall	Deer	FixedObject OffRaod	Non Collision	Other Animal	Pedestrian	{snow, snowfall} [a]	{rock, stone} [b]	fall	strike
Deer	-0.22	-0.83	-0.18	-0.19	1.01	-0.08	-0.14	-0.10	4.99	-2.85
FixedObjectOffRoad	-2.86	-2.98	-2.70	-2.78	-2.72	-3.76	2.12	-4.06	-3.70	-1.75
Pedestrain	-3.38	-3.95	-3.02	-1.82	-3.00	-2.99	-2.76	2.13	-2.99	-0.87
OtherAnimal	-2.37	-0.78	2.91	-0.94	0.05	-0.69	-0.60	-1.96	-3.55	-1.89
FiexdObjectInRoad	-1.97	-1.78	-1.84	-1.79	-0.17	-0.80	-0.63	-0.67	-0.70	-0.68
NonCollision	-0.59	-0.40	-0.44	-0.44	-0.45	0.90	-0.52	-0.47	-0.44	-1.12
BackedInto	-0.25	-0.26	1.54	-0.12	-0.19	-0.13	-1.91	2.09	-1.92	-2.26
Motocyclist	-2.97	-2.06	-2.27	-1.88	1.06	-2.34	-2.28	-1.84	-3.31	-3.17

[a]{snow, snowfall}: precipitation falling from clouds in the form of ice crystals
[b]{rock, stone}: a lump or mass of hard consolidated mineral matter

annotated set of single- and multi-vehicle reports.

VII. Conclusions and Future Work

Structured data are often paired with unstructured language data, and this combination can be exploited using statistical NLP techniques. We have studied the specific application of quality control for transportation reports and have demonstrated a comprehensive procedure comprising data cleaning, annotation, feature extraction, modeling building, and evaluation. Below, we summarize key conclusions, implications, and points of future work.

A. Conclusions

a) Human reporting errors are unavoidable: The analysis of transportation systems is, in many cases, a human-driven process. As such, errors are likely to be introduced at many points in the analysis pipeline. Our research indicates that, in the domain of traffic accident reporting, errors are introduced starting with the first phase of data collection by reporting officers. We have analyzed a sample of this data and found that approximately 16% of the reports contain internal inconsistencies. We believe these inconsistencies could affect downstream safety and planning analyses; however, a manual review of all reports is not feasible.

b) NLP provides automatic quality control: Natural language processing (NLP) techniques provide a way to reconcile structured data with unstructured text in a principled, automated fashion. We have demonstrated the feasibility of applying state-of-the-art NLP techniques to the process of quality control for transportation reports. Our results indicate significant improvements in detecting inconsistent vehicle accident reports versus two baseline approaches, one of which uses handwritten quality control rules. Our best model achieves an F-measure performance of 84.3%.

c) Practical implications: At the local level, our approach constitutes an additional tool for use by maintainers of the accident report database. More importantly, however, our research has implications for any data collection process that pairs structured data with unstructured text. Currently, the latter serves primarily as an auxiliary information source that must be manually interpreted. Our methodology addresses this analytical gap using established NLP techniques that can be ported/scaled to new types of transportation data. Regardless of the application, our methodology should help improve the quality of collected data and subsequent analyses at minimal long-term cost.

B. Future Work

This work sets the stage for a number of future investigations. First, we are interested in analyzing all fields contained in the crash report database instead of the single *CRASH TYPE* field used in this study. Second, we plan to extend our analyses to multi-vehicle reports. Both of these investigations will test the generalization capabilities of the proposed methodology. Third, we believe more advanced language feature extraction will improve the process. Synonymy and bigram information did not help, but a deeper analysis of the text (e.g., by semantic processing [21]) could reveal important information. In terms of modeling, we believe our logistic regression model is well suited to the task; however, improved feature selection (e.g., floating forward-backward selection [16]) might be beneficial given the fact that our simpler, stepwise approach fared well during evaluation. Lastly, we are interested in identifying other transportation data sources that could benefit from this promising approach.

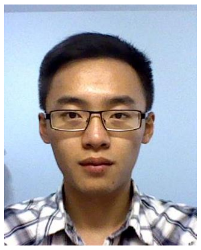
Acknowledgments

This research was funded by a Virginia Department of Transportation (VDOT) Technical Assistance Grant (RC00049). We thank the VDOT personnel for their helpful feedback on our work.

References

[1] N. Garber and A. Ehrhart, "Effect of speed, flow, and geometric characteristics on crash frequency for two-lane highways," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1717, no. -1, pp. 76–83, 2000.
[2] A. MAURINO and C. BATINI, "Methodologies for data quality assessment and improvement," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–52, 2009.
[3] W. E. Winkler, "Methods for evaluating and creating data quality," *Information Systems*, vol. 29, no. 7, pp. 531–550, 2004.
[4] A. Even and G. Shankaranarayanan, "Dual assessment of data quality in customer databases," *Journal of Data and Information Quality (JDIQ)*, vol. 1, no. 3, p. 15, 2009.
[5] B. of Labor Statistics, "Us department of labor," *Occupational Outlook Hankbook, 2012-13 Edition*, Quality Control Inspectors, http://www.bls.gov/ooh/production/quality-control-inspectors.htm (02/12/2013).
[6] J. H. Martin and D. Jurafsky, "Speech and language processing," 2000.
[7] C. Stanfill and B. Kahle, "Parallel free-text search on the connection machine system," *Communications of the ACM*, vol. 29, no. 12, pp. 1229–1239, 1986.
[8] D. Ravichandran, P. Pantel, and E. Hovy, "Randomized algorithms and nlp: Using locality sensitive hash functions for high speed noun clustering," in *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, vol. 43, no. 1, 2005, p. 622.
[9] F. Ciravegna, "Adaptive information extraction from text by rule induction and generalisation," 2001.
[10] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *The Journal of Machine Learning Research*, vol. 3, pp. 1289–1305, 2003.
[11] D. Jeske and R. Liu, "Mining and tracking massive text data: Classification, construction of tracking statistics, and inference under misclassification," *Technometrics*, vol. 49, no. 2, pp. 116–128, 2007.
[12] J. Cohen *et al.*, "A coefficient of agreement for nominal scales," *Educational and psychological measurement*, vol. 20, no. 1, pp. 37–46, 1960.
[13] C. Fellbaum, "Wordnet," *Theory and Applications of Ontology: Computer Applications*, pp. 231–243, 2010.
[14] R. A. Johnson and D. W. Wichern, *Applied multivariate statistical analysis*. Prentice hall Englewood Cliffs, NJ, 1992, vol. 4.
[15] B. Moore, "Principal component analysis in linear systems: Controllability, observability, and model reduction," *Automatic Control, IEEE Transactions on*, vol. 26, no. 1, pp. 17–32, 1981.
[16] P. Pudil, J. Novovicova, and J. Kittler, "Floating search methods in feature selection," *Pattern Recognition Letters*, vol. 15, pp. 1119–1125, 1994.
[17] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin, "LIBLINEAR: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.
[18] B. Ripley, "Nnet: feed-forward neural networks and multinomial log-linear models," 2009.
[19] E. Dimitriadou, K. Hornik, F. Leisch, D. Meyer, A. Weingessel, and M. D. Meyer, "Package e1071," *R Software package*, 2009.
[20] B. Efron and R. Tibshirani, *An introduction to the bootstrap*. Chapman & Hall/CRC, 1994, vol. 57.
[21] M. Gerber and J. Chai, "Beyond NomBank: A study of implicit arguments for nominal predicates," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden: Association for Computational Linguistics, July 2010, pp. 1583–1592. [Online]. Available: http://www.aclweb.org/anthology/P10-1160

Matthew S. Gerber received a Ph.D. in computer science from Michigan State University in 2011 and is currently a Research Assistant Professor in the Department of Systems and Information Engineering at the University of Virginia. His research interests include natural language processing, data mining, and predictive modeling with applications in transportation, crime analysis, and medical informatics.



Lu Tang received a B.A. degree in mathematics and statistics from the University of Virginia in 2012, where he is currently working toward an M.S. degree in the Department of Statistics. Since June 2012, he has been a Research Assistant in the Department of Systems and Information Engineering. His research interests include machine learning, statistical computing, natural language processing, and intelligent transportation systems.