Detecting Errors within a Corpus using Anomaly Detection

Eleazar Eskin
Department of Computer Science
Columbia University
eeskin@cs.columbia.edu

Abstract

We present a method for automatically detecting errors in a manually marked corpus using anomaly detection. Anomaly detection is a method for determining which elements of a large data set do not conform to the whole. This method fits a probability distribution over the data and applies a statistical test to detect anomalous elements. In the corpus error detection problem, anomalous elements are typically marking errors. We present the results of applying this method to the tagged portion of the Penn Treebank corpus.

1 Introduction

Manually marking corpora is a time consuming and expensive process. The process is subject to human error by the experts doing the marking. Unfortunately, many natural language processing methods are sensitive to these errors. In order to ensure accuracy in a corpus, typically several experts pass over the corpus to ensure consistency. For large corpora this can be a tremendous expense.

In this paper, we propose a method for automatically detecting errors in a marked corpus using an anomaly detection technique. This technique detects anomalies or elements which do not fit in with the rest of the corpus. When applied to marked corpora, the anomalies tend to be errors in the markings of the corpus.

To detect the anomalies, we first compute a probability distribution over the entire corpus. Then we apply a statistical test which identifies which elements are anomalies. In this case the anomalies are the elements with very low likelihood. These elements are marked as errors and are thrown out of the corpus. The model is recomputed on the remaining elements. At conclusion, we are left with two data sets: one the normal elements and the second the detected anomalous elements.

We evaluate this method over the part of speech tagged portion of the Penn Treebank corpus (Marcus et al., 1993). In one experiment, our method detected 1000 anomalies within a data set of 1.25 million tagged elements. Human judges evaluated the results of the application of this method and verified that 69% of identified anomalies are in fact tagging errors. In another experiment, our method detected 4000 anomalies of which 44% are tagging errors.

1.1 Related Work

The tagged portion of the Penn Treebank has been extensively utilized for construction and evaluation of taggers. This includes transformation-based tagging (Brill, 1994; Brill and Wu, 1998). Weischedel et al. (1993) applied Markov Models to tagging. Abney et al. (1999) applied boosting to part of speech tagging. Adwait Ratnaparkhi (1996) estimates a probability distribution for tagging using a maximum entropy approach.

Regarding error detection in corpora, Ratnaparkhi (1996) discusses inconsistencies in the Penn Treebank and relates them to inter-annotator differences in tagging style. Abney, Schapire and Singer (1999) discuss how to use boosting for cleaning data.

Much related work to the anomaly detection problem stems from the field of statistics in the study of outliers. This work examines detecting and dealing with outliers in univariate data, multivariate data, and structured data where the probability distribution over the data is given a priori. Statistics gives a set of discordancy tests which can be applied to any given element in the dataset to determine whether it is an outlier. A survey of outliers in statistics is

given in Barnett and Lewis (1994).

Anomaly detection is extensively used within the field of computer security specifically in intrusion detection (Denning, 1987). Typically anomaly detection methods are applied to detect attacks by comparing the activity during an attack to the activity under normal use (Lane and Brodley, 1997; Warrender et al., 1999). The method used in this paper is based on a method for anomaly detection which detects anomalies in noisy data (Eskin, 2000).

The sparse Markov transducer probability modeling method is an extension of adaptive mixtures of probabilistic transducers (Singer, 1997; Pereira and Singer, 1999). Naive Bayes learning, which is used to estimate probabilities in this paper, is described in (Mitchell, 1997).

2 Anomaly Detection

More formally, *anomaly detection* is the process of determining when an element of data is an outlier. Given a set of training data without a probability distribution, we want to construct an automatic method for detecting anomalies. We are interested in detecting anomalies for two main reasons. One, we are interested in modeling the data and the anomalies can contaminate the model. And two, the anomalies themselves can be of interest as they may show rarely occurring events. For the purposes of this work, we are most interested in identifying mistagged elements, i.e. the second case.

In order to motivate a method for detecting anomalies, we must first make assumptions about how the anomalies occur in the data. We use a "mixture model" for explaining the presence of anomalies, one of several popular models in statistics for explaining outliers (Barnett and Lewis, 1994). In the mixture model, there are two probability distributions which generate the data. An element x_i is either generated from the majority distribution or with (small) probability λ from an alternate (anomalous) distribution. Our distribution for the data, $\mathbf{D}$, is then:

$$\mathbf{D} = (1-\lambda)\mathbf{M} + \lambda\mathbf{A} \tag{1}$$

where $\mathbf{M}$ is the majority distribution, and $\mathbf{A}$ is the anomalous distribution. The mixture framework for explaining anomalies is independent of the properties of the distributions $\mathbf{M}$ and $\mathbf{A}$. In other words, no assumptions about the nature of the probability distributions are necessary. The specific probability distributions, $\mathbf{M}$ and $\mathbf{A}$, are chosen based on prior knowledge of the problem. Typically $\mathbf{M}$ is a structured distribution which is estimated over the data using a machine learning technique, while $\mathbf{A}$ is a uniform (random) distribution representing elements which do not fit into $\mathbf{M}$.

In the corpus error detection problem, we are assuming that for each tag in the corpus with probability $(1-\lambda)$ the human annotator markes the corpus with the correct tag and with probability λ the human annotator makes an error. In the case of an error, we assume that the tag is chosen at random.

2.1 Detection of Anomalies

Detecting anomalies, in this framework, is equivalent to determining which elements were generated by the distribution $\mathbf{A}$ and which elements were generated by distribution $\mathbf{M}$. Elements generated by $\mathbf{A}$ are anomalies, while elements generated by $\mathbf{M}$ are not. In our case, we have probability distributions associated with the distributions $\mathbf{M}$ and $\mathbf{A}$, P_M and P_A respectively.

The algorithm partitions the data into two sets, the normal elements M and the anomalies A. For each element, we make a determination of whether it is an anomaly and should be included in A or a majority element in which it should be included in M. We measure the likelihood of the distribution under both cases to make this determination.

The likelihood, L, of distribution $\mathbf{D}$ with probability function P over elements $x_1,...,x_N$ is defined as follows:

$$L(\mathbf{D}) = \prod_{i=1}^{N} P_D(x_i) = \tag{2}$$
$$\left((1-\lambda)^{|M|} \prod_{x_i \in M} P_M(x_i)\right)\left(\lambda^{|A|} \prod_{x_j \in A} P_A(x_j)\right)$$

Since the product of small numbers is difficult to compute, we instead compute the log likelihood, LL. The log likelihood for our case is:

$$LL(\mathbf{D}) = |M|\log(1-\lambda) + \sum_{x_i \in M} \log(P_M(x_i)) + |A|\log\lambda + \sum_{x_j \in A} \log(P_A(x_j)) \tag{3}$$

In order to determine which elements are anomalies, we use a general principal for determining outliers in multivariate data (Barnett, 1979). We measure how likely each element x_i is an outlier by comparing the difference between the log likelihood of the distribution if the element is removed from the majority distribution and included in the anomalous distribution. If this difference is sufficiently large, we declare the element an anomaly.

Specifically what this difference should be depends on the probability distributions and prior knowledge of the problem such as the rate of the anomalies, λ.

3 Methodology

3.1 Corpus

The corpus we use is the Penn Treebank tagged corpus. The corpus contains approximately 1.25 million manually tagged words from Wall Street Journal articles. For each word, a record is generated containing the following elements:

1. The tag of the current word T_i.
2. The current word W_i.
3. The previous tag T_{i-1}.
4. The next tag T_{i+1}.

Over records containing these 4 elements, we compute our probability distributions.

3.2 Probability Modeling Methods

The anomaly detection framework is independent of specific probability distributions. Different probability distributions have different properties. Since the anomaly detection framework does not depend on a specific probability distribution, we can choose the probability distribution to best model the data based on our intuitions about the problem.

To illustrate this, we perform two sets of experiments, each using a different probability distribution modeling method. The first set of experiments uses sparse Markov transducers as the probability modeling method, while the second uses a simple naive Bayes method.

3.3 Sparse Markov Transducers

Sparse Markov transducers compute probabilistic mappings over sparse data. A Markov transducer is defined to be a probability distribution conditional on a finite set of inputs. A Markov transducer of order L is the conditional probability distribution of the form:

$$P(Y_t|X_tX_{t-1}X_{t-2}X_{t-3}...X_{t-(L-1)}) \quad (4)$$

where X_k are random variables over the input alphabet Σ_{in} and Y_k is a random variable over the output alphabet Σ_{out}. This probability distribution stochastically defines a mapping of strings over the input alphabet into the output alphabet. The mapping is conditional on the L previous input symbols.

In the case of sparse data, the probability distribution is conditioned on only some of the inputs. We use sparse Markov transducers to model these type of distributions. A sparse Markov transducer is a conditional probability of the form:

$$P(Y_t|\phi^{n_1}X_{t_1}\phi^{n_2}X_{t_2}...\phi^{n_k}X_{t_k}) \quad (5)$$

where ϕ represents a wild card symbol and $t_i = t - \sum_{j=1}^{i} n_j - (i-1)$. The goal of the sparse Markov transducer estimation algorithm is to estimate a conditional probability of this form based upon a set of inputs and their corresponding outputs. However, the task is complicated due to the lack of knowledge *a priori* of which inputs the probability distribution is conditional on.

Intuitively, a fixed order Markov Chain of order L is equivalent to a n-gram with $n = L$. In a variable order Markov Chain, the value of n changes depending on the context. For example, some elements in the data may use a bigram, while others may use a trigram. The sparse Markov transducer uses a weighted sum of n-grams for different values of n and these weights depend on the context. In addition the weighted sum is over not only n-grams, but also n-grams with wild cards such as a trigram where only the first and last element is conditioned on.

In this case we are looking at the input sequence of the current word, W_t, the previous tag, T_{t-1}, and the next tag, T_{t+1}. The output is the set of all possible tags. The models that are in the weighted sum are the trigram, $W_tT_{t-1}T_{t+1}$; the bigrams W_tT_{t-1}, W_tT_{t+1} and $T_{t-1}T_{t+1}$; and the unigrams W_t, T_{t-1} and T_{t+1}. The specific weights of each model depends on the context or the actual values of W_t, T_{t-1}, and T_{t+1}.

Sparse Markov transducers depend on a set of prior probabilities that incorporate prior knowledge about the importance of various elements in the input sequence. These prior probabilities are set based on the problem. For this problem, we use the priors to encode the information that the current word, W_t, is very important in determining the part of speech.

Each model in the weighted sum uses a pseudo-count predictor. This predictor computes the probability of an output (tag) by the number of times that a specific output was seen in a given context. In order to avoid probabilities of 0, we assume that we have seen each output at least once in every context. In fact, these predictors can be any probability distribution which can also depend on what works best for the task.

3.4 Naive Bayes

The probability distribution for the tags was also estimated using a straight forward naive Bayes approach.

We are interested in the probability of a tag, given the current word, the previous tag, and the next tag, or the probability distribution $P(T_i|W_i, T_{i-1}, T_{i+1})$ which using Bayes Rule is equivalent to:

$$P(T_i|W_i, T_{i-1}, T_{i+1}) = \frac{P(W_i, T_{i-1}, T_{i+1}|T_i) * P(T_i)}{P(W_i, T_{i-1}, T_{i+1})} \quad (6)$$

If we make the Naive Bayes independence assumption and we assume that the denominator is constant for all values this reduces to:

$$P(T_i|W_i, T_{i-1}, T_{i+1}) = \frac{P(W_i|T_i) * P(T_{i-1}|T_i) * P(T_{i+1}|T_i) * P(T_i)}{C} \quad (7)$$

where C is a normalization constant in order to have the probabilities sum to 1. Each of the values on the right side of the equation can easily be computed over the data estimating a probability distribution.

3.5 Computing Probability Distributions

Each probability distribution was trained over each record giving a model over the entire data. The probability model is then used to determine whether or not an element is an anomaly by applying the test in equation (3). Typically this can be done in an efficient manner because the approach does not require reestimating the model over the entire data set. If an element is designated as an anomaly, we remove it from the set of normal elements and efficiently reestimate the probability distribution to obtain more anomalous elements.

4 Results/Evaluation

The method was applied to the Penn Treebank corpus and a set of anomalies were generated. These anomalies were evaluated by human judges to determine if they are in fact tagging errors in the corpus. The human judges were natural language processing researchers (not the author) familiar with the Penn Treebank markings.

In the experiments involving the sparse Markov transducers, after applying the method, 7055 anomalies were detected. In the experiments involving the naive Bayes learning method, 6213 anomalies were detected.

Sample output from the system is shown in figure 1. The error is shown in the context marked with !!!. The likelihood of the tag is also given which is extremely low for the errors. The system also outputs a suggested tag and its likelihood which is the tag with the highest likelihood for that context. As we can see, these errors are clearly annotation errors.

Since the anomalies detected from the two probability modeling methods differed only slightly, we performed human judge verification of the errors over only the results of the sparse Markov transducer experiments.

The anomalies were ordered based on their likelihood. Using this ranking, the set of anomalies were broken up into sets of 1000 records. We examined the first 4000 elements by randomly selecting 100 elements out of each 1000.

Human judges were presented with the system output for four sets of 100 anomalies. The judges were asked to choose among three options for each example:

1. *Corpus Error* - The tag in the corpus sentence is incorrect.

2. *Unsure* - The judge is unsure whether or not the corpus tag is correct.

Error 0.000035: Its/PRP$ fast-food/NN restaurants/NNS -/: including/VBG Denny/NNP 's/POS ,/, Hardee/NNP 's/POS ,/, Quincy/NNP 's/POS and/CC El/NNP Pollo/NNP Loco/NNP (/("/" !!!the/NN!!! only/JJ significant/JJ fast-food/NN chain/NN to/TO specialize/VB in/IN char-broiled/JJ chicken/NN "/")/) -/: are/VBP stable/JJ ,/, recession-resistant/JJ and/CC growing/VBG ./.
Suggested Tag: DT (0.998262)

Error 0.019231: Not/RB even/RB Jack/NNP Lemmon/NNP 's/POS expert/JJ doddering/JJ !!!makes/NNS!!! this/DT trip/NN worth/NN taking/VBG ./.
Suggested Tag: VBZ (0.724359)

Error 0.014286: It/PRP also/RB underscores/VBZ the/DT difficult/JJ task/NN ahead/RB as/IN !!!Coors/NNS!!! attempts/VBZ to/TO purchase/VB Stroh/NNP Brewery/NNP Co./NNP and/CC fight/VB off/RP increasingly/RB tough/JJ competition/NN from/IN Anheuser-Busch/NNP Cos/NNP ./.
Suggested Tag: NNP (0.414286)

Figure 1: Sample output of anomalies in Penn Treebank corpus. The errors are marked with !!!.

3. *System Error* - The tag in the corpus sentence is correct and the system incorrectly marked it as an error.

The "unsure" choice was allowed because of the inherent subtleties in differentiating between types of tags such as "VB vs. VBP" or "VBD vs. VBN".

Over the 400 examples evaluated, 158 were corpus errors, 202 were system errors and the judges were unsure in 40 of the cases. The corpus error rate was computed by throwing out the unsure cases and computing:

$$\text{Corpus error rate} = \frac{\text{Corpus Errors}}{\text{System Errors} + \text{Corpus Errors}} \quad (8)$$

The total corpus error rate over the 400 manually checked examples was was 44%. As can be seen, many of the anomalies are in fact errors in the corpus.

For each error, we asked the human judge to determine if the correct tag is the systems suggested tag. Out of the total 158 corpus errors, the systems correct tag would have corrected the error in 145 cases.

Since the verified examples were random, we can assume that 91% of corpus errors would be automatically corrected if the system would replace the suspect tag with the suggested tag. Ignoring the "unsure" elements for the purposes of this analysis, if we attempted to automatically correct the first 1000 examples where the error rate was 69%, this method would have led to a reduction of the total number of errors in the corpus by 245.

5 Conclusion

This paper presents a fully automatic method for detecting errors in corpora using anomaly detection techniques. As shown, the anomalies detected in the Penn Treebank corpus tend to be tagging errors.

This method has some inherent limitations because not all errors in the corpus would manifest themselves as anomalies. In infrequent contexts or ambiguous situations, the method may not have enough information to detect an error. In addition, if there are inconsistencies between annotators, the method would not detect the errors because the errors would be manifested over a significant portion of the corpus.

Although this paper presents a fully automatic method for error detection in corpora, this method can also be used as a semi-automatic method for correcting errors. The method can guide an annotator to the elements which are most likely errors. The method can greatly reduce the number of elements that an annotator needs to examine.

Future work in this area involves modeling the corpora with other probability distributions.

Anomaly Rank	*Corpus Errors*	*System Error*	*U*nsure	*Corpus Error Rate*
1-1000	63	28	9	69%
1001-2000	36	54	10	40%
2001-3000	18	70	12	20%
3001-4000	41	50	9	45%
Totals	158	202	40	44%

Table 1: Results of error detection experiments on the tagged portion of the Penn Treebank

The method is very sensitive to the effectiveness of the probability model in modeling the normal elements. Extensions to the probability distributions presented here such as adding information about endings of words or using more features could increase the accuracy of the probability distribution and the overall performance of the anomaly detection system. Other future work involves applying this method to other marked corpora.

References

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

V. Barnett and T. Lewis. 1994. *Outliers in Statistical Data.* John Wiley and Sons.

V. Barnett. 1979. Some outlier tests for multivariate samples. *South African Statist*, 13:29–52.

Eric Brill and Jun Wu. 1998. Classifier combination for improved lexical disambiguation. In *Proceedings of COLING-ACL.*

Eric Brill. 1994. Some advances in transformation-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 722–727.

D.E. Denning. 1987. An intrusion detection model. *IEEE Transactions on Software Engineering*, SE-13:222–232.

Eleazar Eskin. 2000. Anomaly detection over noisy data using learned probability distributions. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000) (to appear).*

T. Lane and C. E. Brodley. 1997. Sequence matching and learning in anomaly detection for computer security. In *AAAI Workshop: AI Approaches to Fraud Detection and Risk Management*, pages 43–49. AAAI Press.

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

Tom Mitchell. 1997. *Machine Learning.* McGraw Hill.

Fernando Pereira and Yoram Singer. 1999. An efficient extension to mixture techniques for prediction and decision trees. *Machine Learning*, 36(3):183–199.

Adwait Ratnaparkhi. 1996. A maximum entropy model part-of-speech tagger. In *Proceedings of the Empirical Methods in Natural Language Processing Conference.*

Yoram Singer. 1997. Adaptive mixtures of probalistic transducers. *Neural Computation*, 9(8):1711–1733.

Christina Warrender, Stephanie Forrest, and Barak Pearlmutter. 1999. Detecting intrusions using system calls: alternative data models. In *1999 IEEE Symposium on Security and Privacy*, pages 133–145. IEEE Computer Society.

Ralph Weischedel, Marie Meteer, Richard Schwartz, Lance Ramshaw, and Jeff Palmucci. 1993. Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics*, 19(2):359–382.