 CMC, vol.57, no.1, pp.1-9, 2018

Analyzing Cross-domain Transportation Big Data of New York City with Semi-supervised and Active Learning

Huiyu Sun[1, *] and Suzanne McIntosh[1]

Abstract: The majority of big data analytics applied to transportation datasets suffer from being too domain-specific, that is, they draw conclusions for a dataset based on analytics on the same dataset. This makes models trained from one domain (e.g. taxi data) applies badly to a different domain (e.g. Uber data). To achieve accurate analyses on a new domain, substantial amounts of data must be available, which limits practical applications. To remedy this, we propose to use semi-supervised and active learning of big data to accomplish the domain adaptation task: Selectively choosing a small amount of datapoints from a new domain while achieving comparable performances to using all the datapoints. We choose the New York City (NYC) transportation data of taxi and Uber as our dataset, simulating different domains with 90% as the source data domain for training and the remaining 10% as the target data domain for evaluation. We propose semi-supervised and active learning strategies and apply it to the source domain for selecting datapoints. Experimental results show that our adaptation achieves a comparable performance of using all datapoints while using only a fraction of them, substantially reducing the amount of data required. Our approach has two major advantages: It can make accurate analytics and predictions when big datasets are not available, and even if big datasets are available, our approach chooses the most informative datapoints out of the dataset, making the process much more efficient without having to process huge amounts of data.

Keywords: Big data, taxi and uber, domain adaptation, active learning, semi-supervised learning.

1 Introduction

New York City (NYC) transportation data is available to the public on a number of different domains: Taxi, Uber, etc. and many works [Deri, Franchetti and Moura (2016); Freire, Silva, Vo et al. (2014); Sun and McIntosh (2016); Poulsen, Dekkers, Wagennar et al. (2016); Aslam, Lim and Pan (2012); Cramer and Krueger (2016); Sun, Hu, McIntosh et al. (2018)] have been proposed to take advantage of it to perform various analytics and tasks such as predicting traffic, classifying datasets and offering services. But most works rely on data from the same domain to draw conclusions. And when the analytics and models are applied to a different domain, often the results are not very promising. This is due to variations between data sets leading to difference in the trained models. It is difficult to accurately adapt the model trained on one datasets (domain) to a different dataset without

[1] Computer Science Department, New York University, New York, NY 10012, USA.

[*] Corresponding Author: Huiyu Sun. Email: hs2879@nyu.edu.

training instances available.

Many works investigated learning approaches on big data using a machine learning or neural network based model [Kasun, Zhou, Huang et al. (2013); Najafabadi, Villanustre, Khoshgoftaar (2015); Chen and Lin (2014); Wu, Zhu, Wu et al. (2014); Suthaharan (2014); Belkin, Matveeva and Niyogi (2004); Lv, Duan, Kang et al. (2015)]. Most of these analytics are a form of supervised learning, which requires large quantities of data.

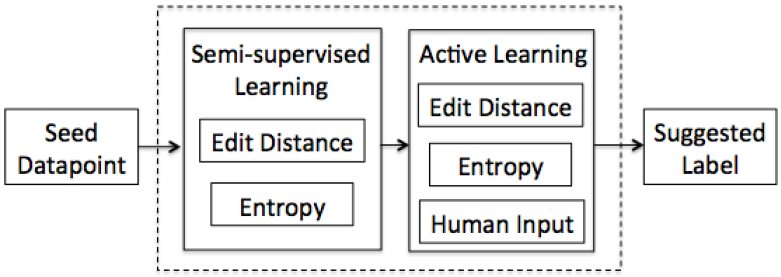

Figure 1: Processing pipeline of our semi-supervised and active learning approach

By contrast, active learning methods [Sun, Grishman and Wang (2017); Akbik, Thilo and Christoph (2014); Sun and Grishman (2018); Sun, Grishman and Wang (2016)] aim to achieve a high performance with as little labeling effort as possible. Semi-supervised [Deri, Franchetti and Moura (2016)] and active learning allow us to selectively choose informative datapoints on the new domain, and using the selected data, it hopes to maximize the performance compared to the performance achieved using all the datapoints on the target domain.

The features for our model is basically the data entries for a given data point: VendorID, pick-up date time, drop-off date time, passenger count, trip distance, pick-up longitude, pick-up latitude, RateCodeID, store and forward flag, drop-off longitude, drop-off latitude, payment type, fare amount, extra, mta tax, tip amount, tolls amount, improvement surcharge, and total amount. Common entries for Uber and taxi are used as features. Variations between different datasets are distinguishable, as found out. Therefore, the features of datasets are enough to distinguish them. The label assigned to each data point is simply which transportation dataset it belongs to: Yellow taxi, Green taxi, or Uber. We investigate domain adaptation of datasets on NYC transportation data: Yellow taxi, Green taxi and Uber.

We use a semi-supervised bootstrapper to classify taxi and Uber data. Bootstrapping is proposed and has been extensively applied. Then active learning is used to select informative datapoints on the target domain. First, we take a semi-supervised approach by investigating ways to bootstrap new data. Then after the most optimized bootstrapping system is achieved, we turn our system into an active learner to select positive and negative datapoints at each iteration. The selected datapoints are given a label (yellow taxi, green taxi or Uber) and are used to classify the target domain data. The pipeline of our system is shown in Fig. 1.

2 Semi-supervised learning

In Semi-supervised learning, we use a bootstrapper that starts out with a set of seed datapoints. It then selects new data based on different sampling strategies. We first consider the edit distance strategy. Each taxi and Uber data entry can be viewed essentially as a string of characters containing time, location, fare amount, etc. The minimum edit distance [Levenshtein (1966)] between two strings (datapoints) is the minimum number of editing operations required to transform one string into the other. The generalized Levenshtein algorithm [Damerau (1964); Bard (2007); Brill and Moore (2000)] with weights and backtrace is used to compute the edit distances. The edit distance scores are normalized to fall in the range of [0, 1] with smaller scores indicating more similar datapoints. Given a set of seed datapoints, we computer the edit distances from a candidate datapoint to each of the seed datapoint. This set of edit distances form an edit distance vector, E, for a candidate datapoint. Then Eq. (1) is applied to E to give a final score, B(x), for a candidate datapoint. This is computed for each datapoint. The datapoints with the smallest scores are selected and added to the seed data set for the next iteration.

$$B(x) = -\sum_{x \in E} x log(x) \quad (1)$$

This edit distance approach is used when bootstrapping. However, another approach is used for comparison: The measure of entropy. Entropy has been used in many works [Zhu, Wang, Yao et al. (2008); Becker, Hachey, Alex et al. (2005)] to measure uncertainty [Lewis and William (1994)] When negative data are introduced, there are two labels for a datapoint: '+' if it is an example of the relation, and '-' if it is not. Then the entropy of a datapoint can be computed by Eq. (2), where P(+) is the probability of a datapoint being labeled positive and P(-) is the probability of a datapoint being labeled negative.

$$B(x) = P(+)logP(+) + P(-)logP(-) \quad (2)$$

For bootstrapping, we pick the candidate data with the smallest entropy, as they are the most reliable data. For active learning, on the other hand, candidate data with the largest entropy are picked since they represent the highest uncertainty.

3 Active learning

An active learner starts out with a set of seed data. It then selects new seed data based on different sampling strategies. The basic idea is to select informative examples that can maximum the likelihood to get the optimal performance. We investigate several different active learning strategies for sample selection.

3.1 Algorithm

The active learning algorithm can be described in Fig. 2. We start with a few seed data. Then the active learning strategy is applied to a set of candidate data and those satisfying the strategy are selected. The iteration continues until a stopping criterion is met. We randomly select 90% of the taxi and Uber dataset as the source domain for training and the rest 10% as the target domain for evaluating our learner. Beside active learning, different search, ranking and information extraction methods [Sun and McIntosh (2016); Sun,

McIntosh and Li (2017); Fu, Ren, Shu et al. (2015); Zhou, Yang, Chen et al. (2016); Fu, Sun, Liu et al. (2015); Yuan, Xia, Sun et al. (2017); Xia, Wang, Sun et al. (2016); Xia, Wang, Sun et al. (2014)] have been proposed.

3.2 Edit distance

An approach to quantifying similarity is using edit distance [Levenshtein (1996)]. The minimum edit distance between two strings is the minimum number of editing operations (insertion, deletion and replacement) required to transform one string into the other. It is being used in speech recognition, computational biology and other fields [Damerau (1964); Bard (2007); Brill and Moore (2000)].

For a datapoint X of length n and Y of length m, we define D(i, j) as the edit distance between X[1..i] and Y[1..j]. The edit distance between X and Y is thus D(n, m). We use the generalized Levenshtein algorithm [Levenshtein (1996); Damerau (1964)] to produce alignments and alignment scores between two datapoints. It uses backtrace to keep track of alignments. The weights to the edit distance are also considered since operations on some datapoint are more likely ranker than others. The algorithm is shown in Fig. 3. With the edit distance approach, the more similar a candidate datapoint is to the seed datapoint, the smaller the edit distance will be. This allows us to rank each candidate datapoint accordingly.

Given:
The set of seed data points, P
The set of candidate data points, C
Sampling methods, $B_1 \ldots B_n$
Size of batch, S

Repeat
  for $s = 1$ **to** S **do**
    // apply sampling strategy
    $x_s^* = argmax_{x \in C} [B_1(x) \cup \ldots \cup B_n(x)]$;
    $P = P \cup (x_s^*)$;
    $C = C - x_s^*$;
  end
Until the predefined stopping criterion is met

Figure 2: Active learning algorithm for selecting new datapoints on the target domain

Given:
$D(0,0) = 0$
$D(i,0) = D(i-1,0) + del[x(j)]; \quad 1 < i \leq N$
$D(0,j) = D(0,j-1) + ins[y(j)]; \quad 1 < j \leq M$

Iterate:
  for $i = 1$ **to** M **do**
    for $j = 1$ **to** N **do**

$$D(i,j) = min \begin{cases} D(i-1,j) + del[x(i)] \\ D(i,j-1) + ins[y(j)] \\ D(i-1,j-1) + rep[x(i), y(j)] \end{cases}$$

    end
  end

Output:
$D(N,M)$ is the distance

Figure 3: Generalized Levenshtein algorithm with weights and backtrace for edit distance

3.3 Improve with entropy

We observe a limitation of our current active learning methods: any particular seed datapoints is limited in that it can only pick out a small set of the available data. Using our active learning methods to quantity similarity between a seed datapoint and potential candidates picks out different ways to describe a specific datapoint. But it sometimes fails to introduce new instances. Here we combine our edit distance methods with entropy to counter the effect of the aforementioned limitation. Given a set of candidates datapoints, first calculate a set of edit distances. Then the edit distance is first divided by the length of the datapoint. This way, long strings (datapoints) will result in an even lower value. Next we calculate the entropy of the candidate set L the same way as semi-supervised learning using Eq. (1).

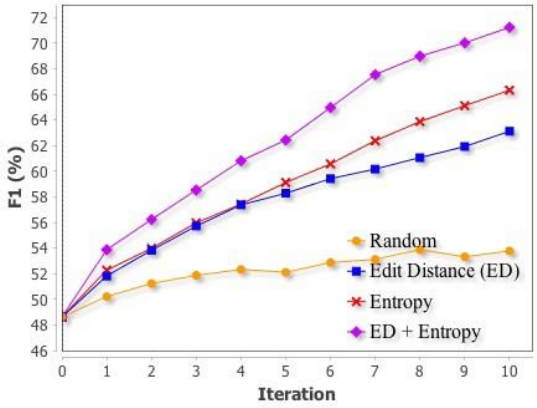

Figure 4: F1 curves for different active learning methods for 10 iterations on the target domain

Entropy has been used in many works to measure uncertainty. Here, we use it as an additional metric in combination with the scores calculated using the edit distance method. The uncertainty part might seem counter-intuitive at first since we are mainly trying to discover similar datapoints. But it does have the benefit in introducing completely different datapoints.

4 Experimental results

4.1 Active learning

We start with a few seed datapoints. The default batch size we used is 1000, meaning in each active learning iteration 1000 new datapoints are selected and added to the data set. We compare the performance in terms of the F-measure achieved, specifically the F1 score. It is a measure of both accuracy and sensitivity as a combination of recall and precision calculated by Eq. (3).

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{3}$$

We compare the performance of edit distance and entropy based active learning methods introduced in the previous section. We also use a random selection strategy for comparison.

Fig. 4 shows the F1 curves for the different methods. Using the 90% split of source domain, we get a F1 of 48.62% of the target domain, which is iteration 0. Then the F1 score improves with each iteration. We performed 10 iterations for each different method.

Randomly selecting datapoints clearly gets much worse performance compared to the other selectively sampling methods. This shows that there is much room for improvement in the sampling method. The Edit distance method, which is the simplest sampling strategy, gets better performance than random by a large margin. The F1 scores after the 10 iterations for the four methods are shown in Tab. 1.

Table 1: F1 scores for random selection and different active learning methods after 10 iterations. Supervised score is also shown

Active Learning Methods	F1 (%)
Random	53.77
Edit Distance	63.14
Entropy	66.32
Edit Distance + Entropy	**71.26**
Supervised	82.37

The entropy active learner has a similar performance compared to edit distance in the first few iterations. But after the 4th iteration, it clearly outperforms edit distance and continues to do so in the later iterations. This shows that as more and more datapoints are selected, entropy is a better method.

As expected, our combination of entropy with edit distance active learner achieved the best performance out of the four, with a better F1 in each iteration of active learning. And after 10 iterations, it beats the next best (entropy) by 5% and beats random selection by 18%. This shows the advantage of leveraging both similarity as well as uncertainty in sample selection.

Using supervised method, i.e. all the datapoints from the 10% target domain, we get a F1 of 82.37%. That is the upper bound that we are trying to compare with using active learning. With edit distance + entropy active learner, we achieved F1 of 71.26 after iterations, 86.52% of supervised score. We only used a fraction of datapoints out of the new domain which contains over millions of data entries. Therefore in terms of data reduction, we used less than 0.01% of data and achieved comparable score to supervised method (86.52% of supervised scores).

4.2 Data splits

To get an overall performance of our active learner, we need to take the split into account. In our experiment, we did a random 90%/10% splits on the train and test data. We want to see the effect that different data split has on the results as some split might pick out a less or more representative data split. We did 20 of these random splits on train and test and recorded the scores on the test. Distribution of the frequency of F1 scores on the test for entropy and edit distance + entropy are shown in Fig. 5. We see that the variation in scores all somewhat conform to a normal distribution.

An overlooked aspect in these big data analytics is the substantial variance in F1 score across different random splits on the dataset as seen in the figure. The standard deviation

(about 2.5% F-measure) is comparable to the difference between systems. To avoid this problem it is necessary to either completely specify the split, so eliminating the variance or take the mean of n splits (thus reducing the SD by a factor of $\sqrt{n}$). In the active learning experiments, we took a split that achieved a score close to the average of the 20 random splits to minimize the variance in scores.

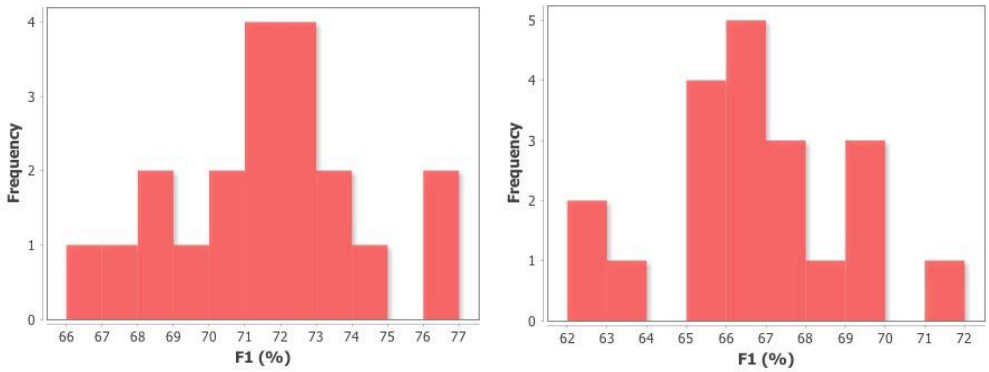

Figure 5: Frequency distributions of F1 scores after 20 random 90%/10% splits on train and test with entropy (*left*) and edit distance + entropy (*right*) active learners

5 Conclusion

In this paper, we have analyzed the possibility of domain adaptation on big datasets and found that an accurate adaptation between the domains of NYC transportation datasets is very feasible. We used semi-supervised learning to train our model, and used active learning to adapt it to a new domain. Using a fraction of labeling effort, we were able to achieve similar performance compared to using all datapoints on the new domain. We used a combination of semi-supervised and active learning to improve the performance of taxi and Uber data. We used semi-supervised learning for training the classification model. We then turned our semi-supervised approach into an active learner, which further increased the performance of our system. The active learner using a combination of edit distance and entropy performed the best. Overall, the process of bootstrapping followed by active learning achieved a F1 score of 71.26% (86.52% of supervised scores) using only a tiny fraction of the dataset, showing domain adaptation of big datasets is able to yield comparable performance to supervised methods.

References

Agichtein, E.; Gravano, L. (2000): Snowball: Extracting relations from large plain-text collections. *Proceedings of the Fifth ACM International Conference on Digital Libraries*, pp. 85-94.

Akbik, A.; Thilo, M.; Christoph, B. (2014): Exploratory relation extraction in large text corpora. *25th International Conference on Computational Linguistics: Technical Papers*, pp. 2087-2096.

Aslam, J.; Lim, S.; Pan, X. (2012): City-scale traffic estimation from a roving sensor

network. *ACM Conference on Embedded Network Sensor Systems*, vol. 42, no. 4, pp. 141-154.

Bard, G. V. (2007): Spelling-error tolerant, order-independent pass-phrases via the Damerau-Levenshtein string-edit distance metric. *Proceedings of the Fifth Australasian Symposium on ACSW Frontiers*, vol. 68, pp. 117-124.

Becker, M.; Hachey, B.; Alex, B. (2005): Optimising selective sampling for bootstrapping named entity recognition. *Proceedings of the Workshop on Learning with Multiple Views*, vol. 2, no. 29, pp. 5-11.

Belkin, M.; Matveeva, I.; Niyogi, P. (2015): Regularization and semi-supervised learning on large graphs. *International Conference on Computational Learning Theory*, pp. 624-638.

Brill, E.; Moore, R. C. (2000): An improved error model for noisy channel spelling correction. *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pp. 286-293.

Chen, X. W.; Lin, X. (2014): Big data deep learning: challenges and perspectives. *IEEE Access*, vol. 2, pp. 514-525.

Cramer, J.; Krueger, A. B. (2016): Disruptive change in the taxi business: The case of Uber. *American Economic Review*, vol. 106, no. 5, pp. 177-182.

Damerau, F. (1964): A technique for computer detection and correction of spelling errors. *Communications of the ACM*, vol. 7, no. 3, pp. 171-176.

Deri, J. A.; Franchetti, F.; Moura, J. M. F. (2016): Big data computation of taxi movement in New York City. *IEEE International Conference on Big Data*, pp. 2616-2625.

Freire, J.; Silva, C.; Vo, H. (2014): Riding from Urban data to insight using New York City Taxis. *IEEE Computer Society Technical Committee on Data Engineering*, vol. 37, no. 4, pp. 43-55.

Fu, Z.; Ren, K.; Shu, J. (2015): Enabling personalized search over encrypted outsourced data with efficiency improvement. *IEEE Transactions on Parallel and Distributed Systems*, vol. 1, no. 1.

Fu, Z.; Sun, X.; Liu, Q. (2015): Achieving efficient cloud search services: Multi-keyword ranked search over encrypted cloud data supporting parallel computing. *IEICE Transactions on Communications*, vol. 1, no. 1, pp. 190-200.

Kasun, L. L. C.; Zhou, H.; Huang, G. B.; Chi, M. V. (2013): Representational learning with ELMs for big data. *Intelligent Systems IEEE*, vol. 28, no. 6, pp. 31-34.

Levenshtein, V. (1996): Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707-710.

Lewis, D. D.; William, A. G. (1994): A sequential algorithm for training text classifiers. *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, vol. 29, no. 2, pp. 3-12.

Lv, Y. S.; Duan, Y. J.; Kang, W. W.; Li, Z. X.; Wang, F. Y. (2015): Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865-873.

Najafabadi, M. M.; Villanustre, F.; Khoshgoftaar, T. (2015): Deep learning applications and challenges in big data analytics. *Journal of Big Data*, vol. 2, no. 1, pp. 1.

Poulsen, L.; Dekkers, D.; Wagennar, N. (2016): Green cabs *vs.* Uber in New York City. *Proceedings of the 2016 IEEE International Congress on Big Data*, pp. 222-229.

Sun, H.; McIntosh, S. (2016): Phone call detection based on smartphone sensor data. *International Conference on Cloud Computing and Security*, vol. 10039, pp. 284-295.

Sun, H.; McIntosh, S. (2016): Big data mobile services for New York City taxi riders and drivers. *Proceedings of the 5th IEEE International Conference on Mobile Services*, pp. 57-64.

Sun, H.; Grishman, R.; Wang, Y. (2017): Active learning based named entity recognition and its application in natural language coverless information hiding. *Journal of Internet Technology*, vol. 18, no. 2, pp. 443-451.

Sun, H.; Grishman, R.; Wang, Y. (2016): Domain adaptation with active learning for named entity recognition. *2nd International Conference on Cloud Computing and Security*, vol. 10040, pp. 611-622.

Sun, H.; McIntosh, S.; Li, B. (2017): Detection of in-progress phone calls using smartphone proximity and orientation sensors. *International Journal of Sensor Networks*, vol. 25, no. 2, pp. 104-114.

Sun, H.; Hu, S.; McIntosh, S. (2018): Big data trip classification on the New York City taxi and Uber sensor network, *Journal of Internet Technology*. (In press)

Suthaharan, S. (2014): Big data classification problems and challenges in network intrusion prediction with machine learning. *ACM Sigmetrics Performance Evaluation Review*, vol. 41, no. 4, pp. 70-73.

Wu, X.; Zhu, X.; Wu, G. Q. (2014): Data mining with big data. *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 1, pp. 97-107.

Xia, Z.; Wang, X.; Sun, X. (2015): A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 340-352.

Xia, Z.; Wang, X.; Sun, X. (2016): Steganalysis of LSB matching using differences between nonadjacent pixels. *Multimedia Tools and Applications*, vol. 75, no. 4, pp. 1947-1962.

Xia, Z.; Wang, X.; Sun, X. (2014): Steganalysis of least significant bit matching using multi-order differences. *Security and Communication Networks*, vol. 7, no. 8, pp. 1283-1291.

Yuan, C.; Xia, Z.; Sun, X. (2017): Coverless image steganography based on SIFT and BOF. *Journal of Internet Technology*, vol. 18, no. 2, pp. 435-442.

Zhou, Z.; Yang, C.; Chen, B. (2016): Effective and efficient image copy detection with resistance to arbitrary rotation. *IEICE Transactions on Information and Systems*, no. 6, pp. 1531-1540.

Zhu, J.; Wang, H.; Yao, T. (2008): Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. *Proceedings of the 22nd International Conference on Computational Linguistics*, vol. 1, pp. 1137-1144.