Standardization of Featureless Variables for Machine Learning Models Using Natural Language Processing

Kourosh Modarresi(✉) and Abdurrahman Munir

Adobe Inc., San Jose, CA, USA
kouroshm@alumni.stanford.edu, munir@adobe.com

Abstract. AI and machine learning are mathematical modeling methods for learning from data and producing intelligent models based on this learning. The data these models need to deal with, is normally a mixed of data type where both numerical (continuous) variables and categorical (non-numerical) data types. Most models in AI and machine learning accept only numerical data as their input and thus, standardization of mixed data into numerical data is a critical step when applying machine learning models. Having data in the standard shape and format that models require often a time consuming, nevertheless very significant step of the process.

Keywords: Machine learning · Natural Language Processing
Mixed type variables

1 Introduction

1.1 Motivation

As an example, when we have a data set (below) combined of many variables where all are numerical ones except two variables of categorical type (gender and marital status) as following [50]:

Table 1. Original mixed variables

User	Age	Income	Gender	Marital status
1	31	90,000	M	Single
2	45	45,000	M	Married
3	63	34,000	M	Divorced
4	33	65,000	F	Divorced
5	47	87,000	F	Single
6	38	39,000	M	Married
7	26	120,000	M	Married
8	25	32,000	F	Married
9	29	55,000	F	Single
10	44	33,000	F	Single

Y. Shi et al. (Eds.): ICCS 2018, LNCS 10861, pp. 234–246, 2018.
https://doi.org/10.1007/978-3-319-93701-4_18

When applying many machine learning models, the models need the data to be numerical data type. Thus, the categorical data should be converted into numerical type. The most efficient way of converting the categorical variable is the introduction of dummy variables (one hot encoding) for which a new (dummy) variable is created for each category (except the last category – since it'd be dependent on the rest of dummy variables, i.e., its value could be determined when all other dummy variables are known) of the categorical variable. These dummy variables are binary variables and could assume only two values, 1 and 0. The value 1 means the sample has the value of that variable and 0 means the opposite.

Here, for this example, we have two categorical variables:

1. Gender: there are only two categories, so we need to create one dummy variable.
2. Marital Status: there are three categories so we need to create two new dummy variables.

The result after the creation of dummy variables is shown in Table 2.

Table 2. The original variables after the introduction of dummy variables.

User	Age	Income	Dummy variable-1 (female)	Dummy variable-2 (married)	Dummy variable-3 (single)
1	31	90000	0	0	1
2	45	45000	0	1	0
3	63	34000	0	0	0
4	33	65000	1	0	0
5	47	87000	1	0	1
6	38	39000	0	1	0
7	26	120000	0	1	0
8	25	32000	1	1	0
9	29	55000	1	0	1
10	44	33000	1	0	1

After this transitional step, we could use any machine learning model for this data set as all its variables are numerical one.

In general, for any categorical variable of "m" categories (classes), we need to create "m − 1" dummy variables. The problem arises when any specific categorical variable has large (based on our work, that means larger than 8) number of categories. The reason is that, in these cases, the number of dummy variables need to be created becomes too large causing the data to become of high dimension. The high dimensionality of data leads to "curse of dimensionality" problem and thus all related issues related to "curse of dimensionality" such as the need of "exponential increase in the number of data rows" and "difficulties of distance computation" would appear. Obviously, one needs to avoid the situation since, in addition to these problems, curse of dimensionality also leads to misleading results from any machine learning models such as finding false patterns discovered based on noise or random chance. Besides all

of that, higher dimension leads to higher "computational cost" and "slow model response and lower robustness", all of which should be avoided. Therefore, in the process of transformation of categorical data into numerical data types, we must reduce the number of newly created numerical variables to reduce the dimension of data [50].

Two examples of the case of categorical variables of large categories or classes are "country of residence" and "URL related data such as the last site visited by the user". For the first variable, there are more than 150 categories and for the second, there is potentially as many categories as the number of users which is a very large (in the order of millions) number. To address these types of problem, this work establishes a new approach of reducing the number of categories (when the number of categories in a categorical variable in larger than 10) to K categories for $K \leq 10$. This way, we will create a limited number of dummy variables to replace the categorical variable in the data set.

For some types of categorical variables such as "country of residence", we may find some attributes online and thus, using these attributes and applying clustering models and web scraping, we can create only a handful of dummy variable to replace the categorical variables of large categories [50].

But, there are other type of categorical variables, such as "URL" variable, where it is not possible to scrap features online and thus the above method [50] cannot be applied. This paper focuses on a method of dealing with this type of categorical data.

2 The Approach Used in This Work

2.1 The Difficulties in Dealing with Modern Data

Quite often, the models in machine learning are models that use only numeric data. Though, practically all data that are used in machine learning are mixed type, numerical and categorical data. When used for machine learning models that could use only numerical data, mixed data types are handled using three different approaches: first approach is trying to, instead, using models that could handle mixed data type, second approach is to ignore (drop) categorical variables. The last approach is converting categorical variables to numerical type by introducing dummy variables. The first approach introduces many limitations as there are only a limited number of models that could handle mixed data and those models are often not the best model fitting the data set. The second approach leads to ignoring much of the information in data set, i.e., the categorical data. The practical approach is the third one, i.e., conversion of categorical data into numerical data. As we explained above, this can be done correctly only when all categorical variables have only limited number of categories (10 or less). Else, it leads to high dimensional data that causes, among other problems, machine learning models to produce meaningless (biased) results. In other words, when the variable has many classes, this approach becomes infeasible because the number of variables will be too much for the numeric models to handle.

This work detects a much smaller number of "latent classes" that are the underpinning classes or categories for the original categories of each categorical variable. This way, the high dimensionality is avoided and thus, we can use these latent classes

to perform the dummy variable generation described above to use any machine learning. The small number of latent categories are detected using k-means clustering.

The basic idea is that categorical variables that have many values (or unique values for each sample) provide little information for other samples. To maintain the useful information from these variables, the best method is to keep that useful (latent) information. This invention does it by finding the latent categories by clustering all categories into similar groups. Using k-means clustering of the categories of any categorical variable, we may two distinct cases. First, is when each category has given features or attributes. This is rarely seen in the data sets. The second case is when there are no such attributes about each of the categories and we need to create them.

In the cases, we have features for all categories or classes of any variable, we could use k-means clustering directly. Though, quite often, there is no attributes information about these classes in the data sets. This work uses NLP [2, 13, 18–20, 53, 57] models (Natural Language Processing) to address the case of categorical variables without any attributes or features. The objective is to find a small number of dummy variables replacing the categorical variable, that we want to convert to a numerical one.

We show our approach for the very important example of URL variable.

2.2 Application of Our Model by Using the Example of URL Data

Categorical variables having URL are important example of these types of categorical variables. They are frequently present in click data and often have very large possible values, sometime as much as the number of users.

To extract the latent categories from these URL variables, we try to cluster them into similar URL's i.e. URLs with similar paths. We choose to extract a word and character using n-gram vector representations from the URL's, then cluster these vector representations using K-means clustering.

URL clustering is a great example because of the difficulty of the task. The difficulty is not only as a result of the number of URLs but also because of the lack of information (attributes) about them that can be used for clustering. When there is no information available about the variables, we need to use NLP. It important that we use NLP to perform the clustering because we have no knowledge of the format of the URLs, i.e., we have no attributions for each URL and clustering cannot be done without attributes. In this case, we use NLP to build the needed attributes for the URLs. When URLs have the same domain, like www.google.com, then the clusters would all be under www.google.com. However, the URLs could also be under multiple domains in which case the clusters would be under multiple domains. A predetermined algorithm would not be able to dynamically handle this variability. This is another reason that, in the case of URLs as an example, we use NLP to cluster them based off syntactic similarity, specifically word bigrams i.e. groups of three words. Our categorical variable has 500 categories, all under the domain of www.adobe.com. A few of these categories are;

http://www.adobe.com/creativecloud.html?promoid=NGWGRLB2&mv=other

http://www.adobe.com/creativecloud/photography.html?promoid=NQCJRBTZ&mv=other

http://www.adobe.com/creativecloud/buy/students.html?promoid=P79NQTWV&mv=other

http://www.adobe.com/creativecloud/business/teams.html?promoid=NYTLR3CX&mv=other

http://www.adobe.com/creativecloud/business/enterprise.html?promoid=NV3KR73Y&mv=other

http://www.adobe.com/creativecloud/buy/education.html?promoid=NLMHRGL1&mv=other

http://www.adobe.com/products/photoshop.html?promoid=PC1PQQ5T&mv=other

http://www.adobe.com/products/illustrator.html?promoid=PGRQQLFS&mv=other

http://www.adobe.com/products/indesign.html?promoid=PLHRQGPR&mv=other

http://www.adobe.com/products/premiere.html?promoid=PQ7SQBYQ&mv=other

http://www.adobe.com/products/experience-design.html?promoid=PYPVQ3HN&mv=other

Fig. 1. The example of URL variable list with 500 different categories.

For the algorithm to work best, we first strip the URL's of any characters that provide little information for clustering (since these words may introduce no new information). These words include punctuation and common words such as "http" and "www". We, thus, perform pre-processing on this list which includes removing punctuation, queries (anything after the character "?"), and stop-words (http, com, www, html, etc.). After this step, we are left with the URLs as space separated words representing the path of URL (Fig. 2);

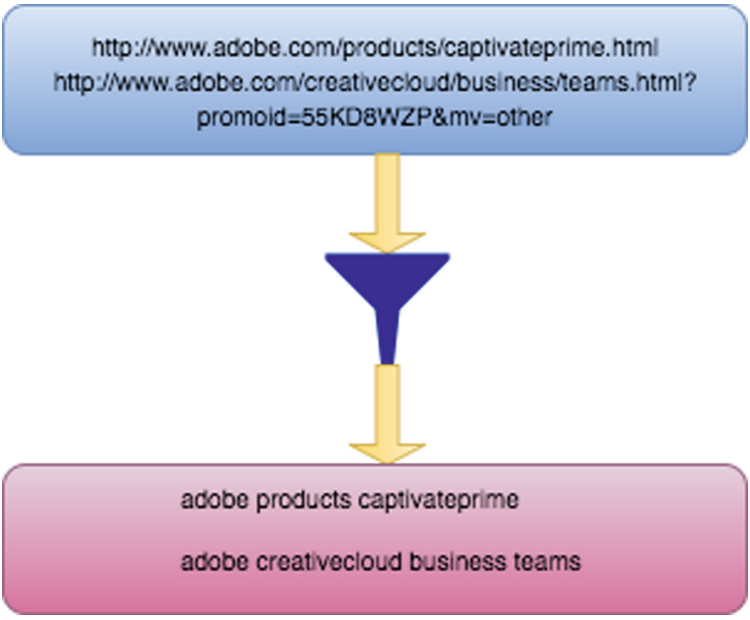

Fig. 2. The process of deleting noisy words from the url variable.

A sample of the result looks like (Fig. 3):

```
adobe creativecloud business teams
 adobe creativecloud desktop-app
 adobe creativecloud business enterprise
 adobe creativecloud business teams
 adobe creativecloud business enterprise
 adobe creativecloud business teams plans
 adobe creativecloud

adobe creativecloud buy students
 adobe creativecloud buy education
 adobe creativecloud buy students
 adobe creativecloud buy students
 adobe creativecloud buy education
 adobe creativecloud buy government
 adobe creativecloud buy government
```

Fig. 3. The url data after the removal of words that may be irrelevant for clustering.

One of the most popular tools in NLP is the ones involving representation of words with a numerical vector representation in an n dimensional space. Using the context of a word, it can be mapped into an n-dimensional vector space. Learned representations such as word embedding is increasingly popular for modeling semantics in NLP. This is done by reducing semantic composition to simple vector operations. We've modified and extended traditional representation learning techniques [13, 18, 50] to support multiple word senses and uncertain representations.

In this work, we used a modification so that, instead of projecting individual words, we project whole URLs containing multiple words. We use these words and their contexts as features for the projection of the whole URL (Fig. 4).

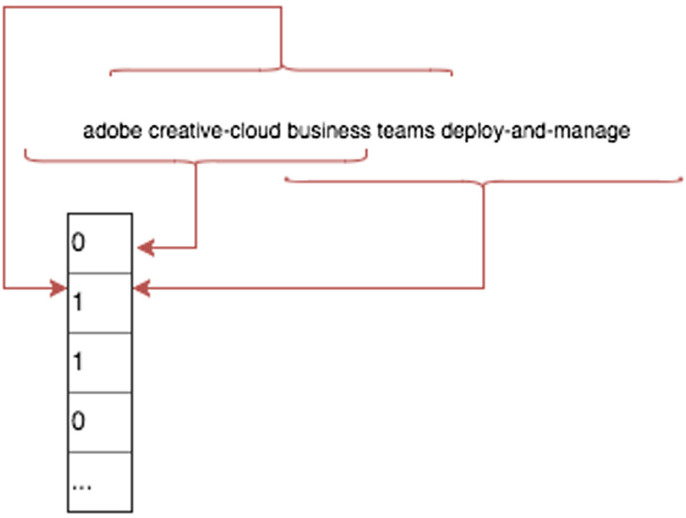

Fig. 4. Vector representation of the url data.

Using the cleaned list, we extract vector representations of the URL's using the tool "Sally". Sally is a tool that maps a set of strings to a set of vectors. The features that we use for this mapping are bi-gram words and tri-gram characters. Thus, using word bigrams of the URLs as features, we project the URLs into vector space using "Sally". Sally represents the URLs using a sparse matrix representation. This means that the URLs are projected into very long vectors with each dimension representing a word trigram that has been seen in the dataset. If a trigram has been observed in the URL its value in the vector is 1. Otherwise the value is 0. This results in a long vector with most values equal to 0 and a few values equal to 1. All the vectors together make a matrix that is a sparse matrix because of its many 0 values. Finally, we used K-means clustering on the embedding. Given that the URLs have been transformed into points in n-dimensional vector space, K-means clustering can find groups of points and partitions them as a cluster in the dataset. Given a number K which is the number of clusters for the algorithm to discover, K-means finds the best partitioning of the dataset such that the points in the clusters are mutually as similar as possible. In the context of URLs this means finding the groups of URLs that share the most word trigrams. Figure 5 shows that the best K values is 10.

Clusters	Word tri-grams	Char tri-grams
3	.149	.224
4	.176	.215
5	.167	.203
6	.172	.199
7	.215	.211
8	.208	.251
9	.210	.244
10	.233	.221

Fig. 5. The computation of optimal number of clustering using word tri-grams.

2.3 Computing the Optimal Number of Clusters

To compute the optimal number of clusters, we use Silhouette method which is based on minimizing the dissimilarities inside a cluster and maximizing the dissimilarities among clusters [31, 50]:

The Silhouette model computes s(i) for each data point in the data set for each K:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Where $a(i)$ is the mean distance of point i to all the other points in its cluster. Also, $b(i)$ is the mean distance to all the points in its closest cluster, i.e., $b(i)$ is the minimum mean distance of point i to all clusters that i is not a member of.

The optimal K is the K that maximizes the total score s(i) for all data set. The score values lie in the range of [−1, 1] with −1 to be the worst possible score and +1 to be the optimal score. Thus, the closest (average score of all points) score to +1 is the optimal one and the corresponding K is the optimal K. Our experiments show that the value of K has upper bound of 10. Here, we use not only the score but the maximum separation and compactness of the clusters, as measured by distance between clusters and uniformity of the width of clusters, to test and validate our model simultaneously when computing optimal K. Figure 6 depicts Silhouette model for different K [50].

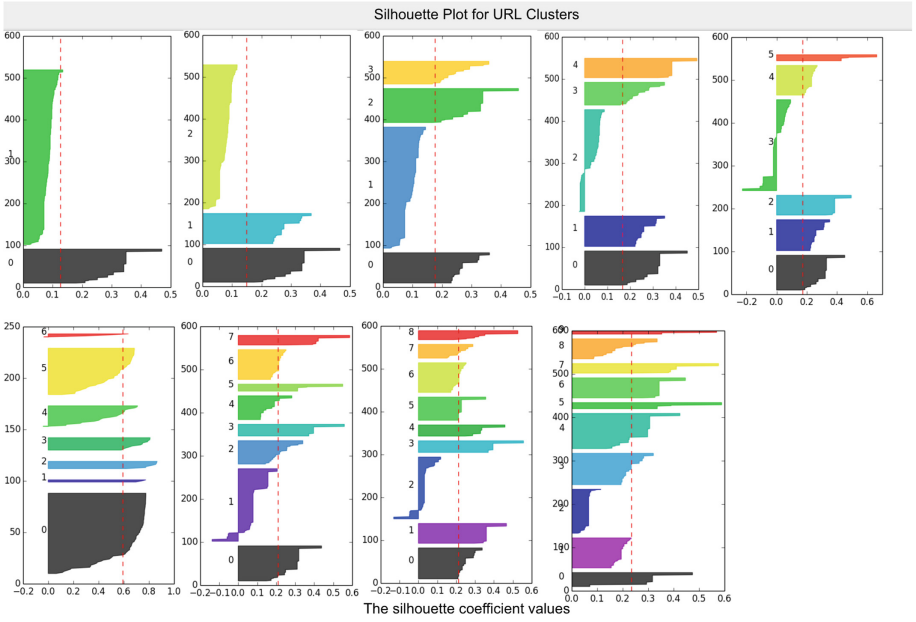

Fig. 6. Using silhouette model to compute the optimal number of clusters, to be 10.

Using the results from silhouette model, we use k-means clustering to cluster the URL data. Some of the clusters are shown in Fig. 7.

```
adobe data-analytics-cloud
 adobe data-analytics-cloud analytics
 adobe data-analytics-cloud
 adobe data-analytics-cloud analytics
 adobe data-analytics-cloud
 adobe data-analytics-cloud
 adobe data-analytics-cloud analytics
 adobe data-analytics-cloud
 adobe data-analytics-cloud analytics
 adobe data-analytics-cloud analytics
 adobe data-analytics-cloud analytics
 adobe data-analytics-cloud analytics select
 adobe data-analytics-cloud analytics prime
 adobe data-analytics-cloud analytics ultimate
 adobe data-analytics-cloud analytics video
 adobe data-analytics-cloud analytics predictive-intelligence
 adobe data-analytics-cloud analytics live-stream
 adobe data-analytics-cloud analytics data-workbench
 adobe data-analytics-cloud analytics mobile-app-analytics
 adobe data-analytics-cloud analytics capabilities
 adobe data-analytics-cloud analytics new-capabilities
 adobe data-analytics-cloud analytics resources
 adobe data-analytics-cloud analytics learn-support
 adobe data-analytics-cloud analytics select
 adobe data-analytics-cloud analytics prime
 adobe data-analytics-cloud analytics ultimate
 adobe data-analytics-cloud analytics video
 adobe data-analytics-cloud analytics predictive-intelligence
 adobe data-analytics-cloud analytics live-stream
 adobe data-analytics-cloud analytics data-workbench
 adobe data-analytics-cloud analytics mobile-app-analytics
 adobe data-analytics-cloud analytics marketing-attribution
 adobe data-analytics-cloud analytics analysis-workspace

adobe products photoshop
 adobe products illustrator
 adobe products indesign
 adobe products premiere
 adobe products experience-design
 adobe products elements-family
 adobe products special-offers
 adobe products photoshop
 adobe products photoshop-lightroom
 adobe products illustrator
 adobe products premiere
 adobe products indesign
 adobe products experience-design
 adobe products captur
```

Fig. 7. Some of the clusters for the url data.

As the figure above shows, our method has grouped together URLs with similar paths and separated URLs with dissimilar paths.

3 The Results and Conclusion

This project provides a method of converting categorical variables to numerical variables so machine learning models could use data. For this conversion to be plausible for categorical variables with many classes, we propose that clustering can be used to decrease the number of classes in the variable to a small number for dummy variable generation. Though, some variables may have accessible features which makes it possible to cluster them, but many variables lack the information or features that would be needed for clustering models. This work deal effectively with these types of categorical variables and assumes no extra features and information may be available, neither explicitly nor implicitly – by web scraping, for such variables. For the model to work, we used NLP to create a vector representation of the variables. Then, we use the vector representation to cluster the variables, i.e., clustering the categories of the variables.

This work provides a new and only practical method of dealing with the standardization of categorical variables when the variables have large number of categories or classes and have no explicitly or implicitly available features. Our model avoids the deletion of the categorical variables and thus loss of information that causes machine learning models to produce meaningless results. This work also leads to the avoidance of creating high dimensional data where "curse of dimensionality" leads to high computational cost, need of exponentially larger data sets, distorted values for distance metrics and biased models.

References

1. Ahn, D., Jijkoun, V., Mishne, G., Müller, K., de Rijke, M., Schlobach, S.: Using Wikipedia at the TREC QA track. In: Proceedings of TREC (2004)
2. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_52
3. Backstrom, L., Leskovec, J.: Supervised random walks: predicting and recommending links in social networks. In: ACM International Conference on Web Search and Data Mining, WSDM (2011)
4. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: International Conference on Learning Representations, ICLR (2015)
5. Baudiš, P.: YodaQA: a modular question answering system pipeline. In: POSTER 2015-19th International Student Conference on Electrical Engineering, pp. 1156–1165 (2015)
6. Baudiš, P., Šedivý, J.: Modeling of the question answering task in the YodaQA system. In: Mothe, J., Savoy, J., Kamps, J., Pinel-Sauvagnat, K., Jones, G.J.F., SanJuan, E., Cappellato, L., Ferro, N. (eds.) CLEF 2015. LNCS, vol. 9283, pp. 222–228. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24027-5_20

7. Becker, S., Bobin, J., Candès, E.J.: NESTA: a fast and accurate first-order method for sparse recovery. SIAM J. Imag. Sci. **4**(1), 1–39 (2009)
8. Bjorck, A.: Numerical Methods for Least Squares Problems. SIAM, Philadelphia (1996)
9. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
10. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1247–1250. ACM (2008)
11. Brill, E., Dumais, S., Banko, M.: An analysis of the AskMSR question-answering system. In: Empirical Methods in Natural Language Processing, EMNLP, pp. 257–264 (2002)
12. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
13. Buscaldi, D., Rosso, P.: Mining knowledge from Wikipedia for the question answering task. In: International Conference on Language Resources and Evaluation, LREC, pp. 727–730 (2006)
14. Candès, E.J., Recht, B.: Exact matrix completion via convex optimization. Found. Comput. Math. **9**, 717–772 (2008)
15. Candès, E.J.: Compressive sampling. In: Proceedings of the International Congress of Mathematicians, Madrid, Spain (2006)
16. Candès, E.J., Tao, T.: Near-optimal signal recovery from random projections: universal encoding strategies. IEEE Trans. Inf. Theory **52**, 5406–5425 (2004)
17. Caruana, R.: Multitask learning. In: Thrun, S., Pratt, L. (eds.) Learning to Learn, pp. 95–133. Springer, Boston (1998). https://doi.org/10.1007/978-1-4615-5529-2_5
18. Chen, D., Bolton, J., Manning, C.D.: A thorough examination of the CNN/Daily Mail reading comprehension task. In: Association for Computational Linguistics, ACL (2016)
19. Chen, D., Fisch, A., Weston, J., Bordes, A.: Reading Wikipedia to answer open-domain questions. arXiv:1704.00051 (2017)
20. Collobert, R., Weston, J.: A unified architecture for natural language processing: deep neural networks with multitask learning. In: International Conference on Machine Learning, ICML (2008)
21. d'Aspremont, A., El Ghaoui, L., Jordan, M.I., Lanckriet, G.R.G.: A direct formulation for sparse PCA using semidefinite programming. SIAM Rev. **49**(3), 434–448 (2007)
22. Efron, B., Hastie, T., Johnstone, I., Tibshirani, R.: Least angle regression. Ann. Stat. **32**, 407–499 (2004)
23. Eldén, L.: Algorithms for the regularization of ill-conditioned least squares problems. BIT **17**, 134–145 (1977)
24. Eldén, L.: A note on the computation of the generalized cross-validation function for ill-conditioned least squares problems. BIT **24**, 467–472 (1984)
25. Engl, H.W., Groetsch, C.W. (eds.): Inverse and Ill-Posed Problems. Academic Press, London (1987)
26. Fader, A., Zettlemoyer, L., Etzioni, O.: Open question answering over curated and extracted knowledge bases. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1156–1165 (2014)
27. Fazel, M., Hindi, H., Boyd, S.: A rank minimization heuristic with application to minimum order system approximation. In: Proceedings American Control Conference, vol. 6, pp. 4734–4739 (2001)
28. Golub, G.H., Van Loan, C.F.: Matrix Computations, 4th edn. Computer Assisted Mechanics and Engineering Sciences, Johns Hopkins University Press, Baltimore (2013)

29. Golub, G.H., Van Loan, C.F.: An analysis of the total least squares problem. SIAM J. Numer. Anal. **17**, 883–893 (1980)
30. Golub, G.H., Heath, M., Wahba, G.: Generalized cross-validation as a method for choosing a good ridge parameter. Technometrics **21**, 215–223 (1979)
31. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning; Data Mining, Inference and Prediction. Springer, New York (2001). https://doi.org/10.1007/978-0-387-21606-5
32. Hastie, T.J., Tibshirani, R.: Handwritten digit recognition via deformable prototypes. Technical report. AT&T Bell Laboratories (1994)
33. Hein, T., Hofmann, B.: On the nature of ill-posedness of an inverse problem in option pricing. Inverse Probl. **19**, 1319–1338 (2003)
34. Hewlett, D., Lacoste, A., Jones, L., Polosukhin, I., Fandrianto, A., Han, J., Kelcey, M., Berthelot, D.: WikiReading: a novel large-scale language understanding task over wikipedia. In: Association for Computational Linguistics, ACL, pp. 1535–1545 (2016)
35. Hill, F., Bordes, A., Chopra, S., Weston, J.: The Goldilocks principle: reading children's books with explicit memory representations. In: International Conference on Learning Representations, ICLR (2016)
36. Hua, T.A., Gunst, R.F.: Generalized ridge regression: a note on negative ridge parameters. Commun. Stat. Theory Methods **12**, 37–45 (1983)
37. Jolliffe, I.T., Trendafilov, N.T., Uddin, M.: A modified principal component technique based on the LASSO. J. Comput. Graph. Stat. **12**, 531–547 (2003)
38. Kirsch, A.: An Introduction to the Mathematical theory of Inverse Problems. Springer, New York (1996). https://doi.org/10.1007/978-1-4419-8474-6
39. Mardia, K., Kent, J., Bibby, J.: Multivariate Analysis. Academic Press, New York (1979)
40. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J., McClosky, D.: The stanford CoreNLP natural language processing toolkit. In: Association for Computational Linguistics, ACL, pp. 55–60 (2014)
41. Marquardt, D.W.: Generalized inverses, ridge regression, biased linear estimation, and nonlinear estimation. Technometrics **12**, 591–612 (1970)
42. Mazumder, R., Hastie, T., Tibshirani, R.: Spectral regularization algorithms for learning large incomplete matrices. JMLR **2010**(11), 2287–2322 (2010)
43. McCabe, G.: Principal variables. Technometrics **26**, 137–144 (1984)
44. Miller, A.H., Fisch, A., Dodge, J., Karimi, A.-H., Bordes, A., Weston, J.: Key-value memory networks for directly reading documents. In: Empirical Methods in Natural Language Processing, EMNLP, pp. 1400–1409 (2016)
45. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: Association for Computational Linguistics and International Joint Conference on Natural Language Processing, ACL/IJCNLP, pp. 1003–1011 (2009)
46. Modarresi, K., Golub, G.H.: An adaptive solution of linear inverse problems. In: Proceedings of Inverse Problems Design and Optimization Symposium, IPDO 2007, Miami Beach, Florida, 16–18 April, pp. 333–340 (2007)
47. Modarresi, K.: A local regularization method using multiple regularization levels, Stanford, CA, April 2007
48. Modarresi, K.: Algorithmic approach for learning a comprehensive view of online users. Proc. Comput. Sci. **80**(C), 2181–2189 (2016)
49. Modarresi, K.: Computation of recommender system using localized regularization. Proc. Comput. Sci. **51**(C), 2407–2416 (2015)
50. Modarresi, K., Munir, A.: Generalized variable conversion using K-means clustering and web scraping. In: ICCS 2018 (2018, Accepted)

51. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: SQuAD: 100,000 + questions for machine comprehension of text. In: Empirical Methods in Natural Language Processing, EMNLP (2016)
52. Ryu, P.-M., Jang, M.-G., Kim, H.-K.: Open domain question answering using Wikipedia-based knowledge model. Inf. Process. Manag. **50**(5), 683–692 (2014)
53. Seo, M., Kembhavi, A., Farhadi, A., Hajishirzi, H.: Bidirectional attention flow for machine comprehension. arXiv preprint arXiv:1611.01603 (2016)
54. Tarantola, A.: Inverse Problem Theory. Elsevir, Amsterdam (1987)
55. Tibshirani, R.: Regression shrinkage and selection via the LASSO. J. Roy. Stat. Soc. Ser. B **58**(1), 267–288 (1996)
56. Tikhonov, A.N., Goncharsky, A.V. (eds.): Ill-Posed Problems in the Natural Sciences. MIR, Moscow (1987)
57. Wang, Z., Mi, H., Hamza, W., Florian, R.: Multi-perspective context matching for machine comprehension. arXiv preprint arXiv:1612.04211 (2016)
58. Witten, R., Candès, E.J.: Randomized algorithms for low-rank matrix factorizations: sharp performance bounds. Algorithmica **72**, 264–281 (2013)
59. Zhou, Z., Wright, J., Li, X., Candès, E.J., Ma, Y.: Stable principal component pursuit. In: Proceedings of International Symposium on Information Theory, June 2010
60. Zou, H., Hastie, T., Tibshirani, R.: Sparse principal component analysis. J. Comput. Graph. Stat. **15**(2), 265–286 (2006)