# Instate: Predicting the State of Residence From Last Name

*

Atul Dhingra[†] Gaurav Sood[‡]

March 12, 2023

**Abstract**

India has twenty-two official languages. Serving such a diverse language base is a challenge for survey statisticians, call center operators, software developers, and other such service providers. To help provide better services to different language communities via better localization, we introduce a new machine learning model that predicts the language(s) that the user can speak from their name. Using nearly 438M records spanning 33 Indian states and 1.13M unique last names from the Indian Electoral Rolls Corpus (Sood and Dhingra, 2023), we build a character-level transformer-based machine-learning model that predicts the state of residence based on the last name. The model has a top-3 accuracy of 81.3% on unseen names. We map the states to languages using the Indian census to infer languages understood by the respondent. We provide open-source software that implements the method discussed in the paper.

**Key words:** localization, machine learning, RNN, LSTM, GRU

## 1 Introduction

An overwhelming majority of people in India don't speak the language in which the vast majority of web content and services are delivered. 57% of the web content is in English.[1] And according to the last Indian census, less than 150M Indians of the more than 1.2B could speak English (and likely much fewer can read it). Hence, localization is essential for delivering software and services. Localization proceeds in two steps: inferring (or letting users select) the language understood by the user and rendering the content or service

*Data and scripts needed to replicate the results are available at: https://github.com/appeler/instate/

[†]Senior Member, IEEE; dhingra.atul92@gmail.com

[‡]Independent researcher; gsood07@gmail.com.

[1]According to this article, English is used by 57% of websites whose language is decipherable.

in that language. In this paper, we present a way to improve localization in India. In particular, we provide a machine-learning model that predicts the languages that are known to the user. To infer the language(s) understood by the user, we rely on names. Using nearly 438M records spanning 33 states and 1.13M unique last names from the Indian Electoral Rolls Corpus (Sood and Dhingra, 2023), we build a character-level machine-learning model that predicts the state of residence based on the last name. The best-performing model has a top-3 accuracy of 85.3% on unseen names. Using the data from the Indian census, we map the states to languages and provide a prediction for the languages understood by the respondent. We also provide open-source software that implements the method discussed in the paper.

## 2 Related Work

Learning sociodemographic characteristics of people, e.g., race and ethnicity, from their names is a well-studied problem (Imai and Khanna, 2016, Parasurama, 2021, Wong et al., 2020, Ye et al., 2017, Sood and Laohaprapanon, 2018). A variety of techniques have been invented for the purpose. For instance, Blevins and Mullen (2015) use a Naive Bayes classifier on historical census data. Imai and Khanna (2016) combine voter registration data with census estimates to infer the race/ethnicity of a person given their last name. Other researchers use embeddings of email and other public social networks to infer ethnicity (Ye et al., 2017, Hur, 2022, Ye and Skiena, 2019). Yet others have used deep learning approaches to exploit the patterns of letters in the name to infer sociodemographic characteristics (Hu et al., 2021, Sood and Laohaprapanon, 2018, Parasurama, 2021). Our paper builds on this extensive body of work to infer what potential languages a person may be able to speak.

## 3 Data

We exploit the Indian electoral rolls data (Sood and Dhingra, 2023, Sood et al., 2018) to build the machine learning model. The corpus includes data on nearly 438M people from 33 states with nearly 1.14 million unique last name spellings. The electoral roll data includes information on the elector's husband or father's name, age, sex, house number, and geographical details, like the polling station, constituency, local police station, etc. Electoral rolls in India are a particularly useful source of data given that they are a proximate census of the adult population. Unlike the US, in India, a neutral administrative body called the Election Commission works to produce an exhaustive list of eligible voters—all adult citizens. The electoral rolls are published online ahead of the elections to make sure that no one is left out, and so that others can point to any errors, e.g., dead voters, missing voters, etc. There are, however, some problems with the data. Primarily, voters are registered in their 'home' districts, even if they are working elsewhere. This works to our

advantage to the extent that we rely on name, and state mappings to infer the languages that the user can understand.

The parsed electoral rolls corpus (Sood et al., 2018) includes transliterations to English using the Python package indicate (Chintalapati and Sood, 2022). There is no unique English transliteration of Hindi names and the package provides the most probable transliteration.

We start by collating the data, preserving only five columns: first name, last name, father's or husband's name, sex, and state. To establish the last name of each person, we work as follows: 1. if the person's name is more than one word, we assume the last word to be the person's last name, 2. else, we check if the father's or husband's name is more than one word and where it is, we assume the last word to be the last name. We discard all other rows as cases where we couldn't establish the last name. We further discard cases where the last name is less than three letters as manual analysis of the data suggests that these are not last names. We also remove last names that have non-alphanumeric characters. Finally, to preserve privacy and because really infrequent last names are unlikely to be true last names, we exclude last names that occur less than thrice across the dataset. Finally, we convert all the last names into lowercase.

## 4 Models

If we have no information from the electoral rolls, then given the last name, the optimal prediction for the state of residence is the most populous state. In our case, that is Uttar Pradesh, which has nearly 230M people. If we did that, we would be right about 16.74% of the time.

If we assume that electoral rolls constitute the universe of people in India with the correct, unique transliterated English spellings of their names, and if the last name is the only piece of information we have about the person, the Bayes Optimal classifier is simply an intercept-only model that gives the population mean—the proportion of last names in various states.

The Naive Bayes model rests on untenable assumptions. It assumes no data entry errors, unique transliterations, and a universe of the adult population. (We know that we have only a fraction of the total adult population of India, which is nearly a billion.)

Hence, a more tenable baseline is the KNN model. To find the optimal K, we start by grouping the data by last name so that for each name we have a 31-state-long vector that tallies the proportion of the last name in each state. We then split the data into a training set (90%), a validation set (5%), and a holdout set (5%). Given the size of the data, rather than use the more conventional but more computationally expensive edit distance measures like the Levenshtein distance, we use a bi-char-based cosine distance as a measure of distance between the names. Then, assuming the training set as our lookup corpus, we find

the closest K states in the training data, take a simple average, and find the modal state. We test K = 3, 5, and 10. We find K of 5 and 10 are equally accurate, predicting the modal state among unseen names correctly about 65% of the time. The striking performance is a result of the fact that there are multiple transliterations of the same name. In effect, for many of the test set entries, we have names with similar distributions in the training set.

Against this competitive baseline, we test three deep learning models—RNN, LSTM, and GRU. The setup for all three models is similar. We split the data by last names and keep 80% of the data for training and 20% for testing. For the test data, we once again group by last names so that we have as many rows as unique names in the test set. For the training data, we transform the data to preserve all the unique names per state, which leaves us with approximately 1.5M rows. For each of the models, we predict the top-3 states and incur a loss if our predictions don't include any of the top-3 states. Before describing the results, we discuss the motivation and setup for each of the models:

- **RNN** We estimate a 2-layer RNN (Wu and He, 2020, Hua et al., 2018) with 512 hidden units, train the model with a batch size of 256 using SGD optimizer and use a negative log-likelihood loss. To deal with exploding gradients, we set a low learning rate of .005, and momentum of 0.9.

- **LSTM** While training the RNN we observed that the training loss was slow to converge, and plateaued after 5000 epochs. To increase the model capacity while avoiding common issues with deep RNN models—exploding or vanishing gradients—we build an LSTM model (Hochreiter and Schmidhuber, 1997). We estimate a 2-layer LSTM with 512 hidden units, train the model for 10000 epochs with a batch size of 256 using the Adam optimizer, and use a negative log-likelihood loss. We set a low learning rate of 3e-4.

- **GRU** One of the drawbacks of using an LSTM is that it is very slow to train and converge. Therefore, to add more capacity to the model, we chose GRU which we could train for longer and converged faster. We estimated a 2-layer GRU (Cho et al., 2014) with 2048 hidden units and trained the model for 5000 epochs with a batch size of 1024 using Adam optimizer with a learning rate of 3e-4 and negative log-likelihood loss.

We comprehensively evaluate the performance of these models. We compare the performance of the models on the test set, a weighted random sample of 3000 unique names with weights in proportion to their popularity, the top 3000 most popular names, and 3000 least popular names. Table 1 summarizes the results. As the table shows, GRU is the best-performing model across the board. The best-performing GRU model has a top-3 accuracy of 85.3% in the test set. Across various slices, the accuracy of the GRU model never dips below 82%.

**Table 1:** Accuracy of Different Models on the Test Set

| Model | Test Set | Weighted Random (3k) | Top-3k | Bottom-3k |
|---|---|---|---|---|
| RNN | 65.3 | 65.4 | 62.7 | 62.3 |
| LSTM | 75.9 | 75.4 | 73.1 | 72.4 |
| GRU | 85.3 | 84.1 | 82.4 | 82.0 |

To dig deeper into the relationship between accuracy and popularity, we plot a lowess between the two across using our random sample. Given the extreme skew in the distribution of last names, we truncate the sample at names shared by 2000 people. As Figure 1 shows, GRU wins handily except at the tail end where we have very limited data.

To shed light on how accuracy may vary by the proportion of women with the last name, we plot a lowess between the two using our random sample. As Figure 2 shows, GRU wins handily.

Lastly, we check how the models perform across states. To check the performance, we take a random sample of 1000 unique names per state. Table 2 summarizes the results.

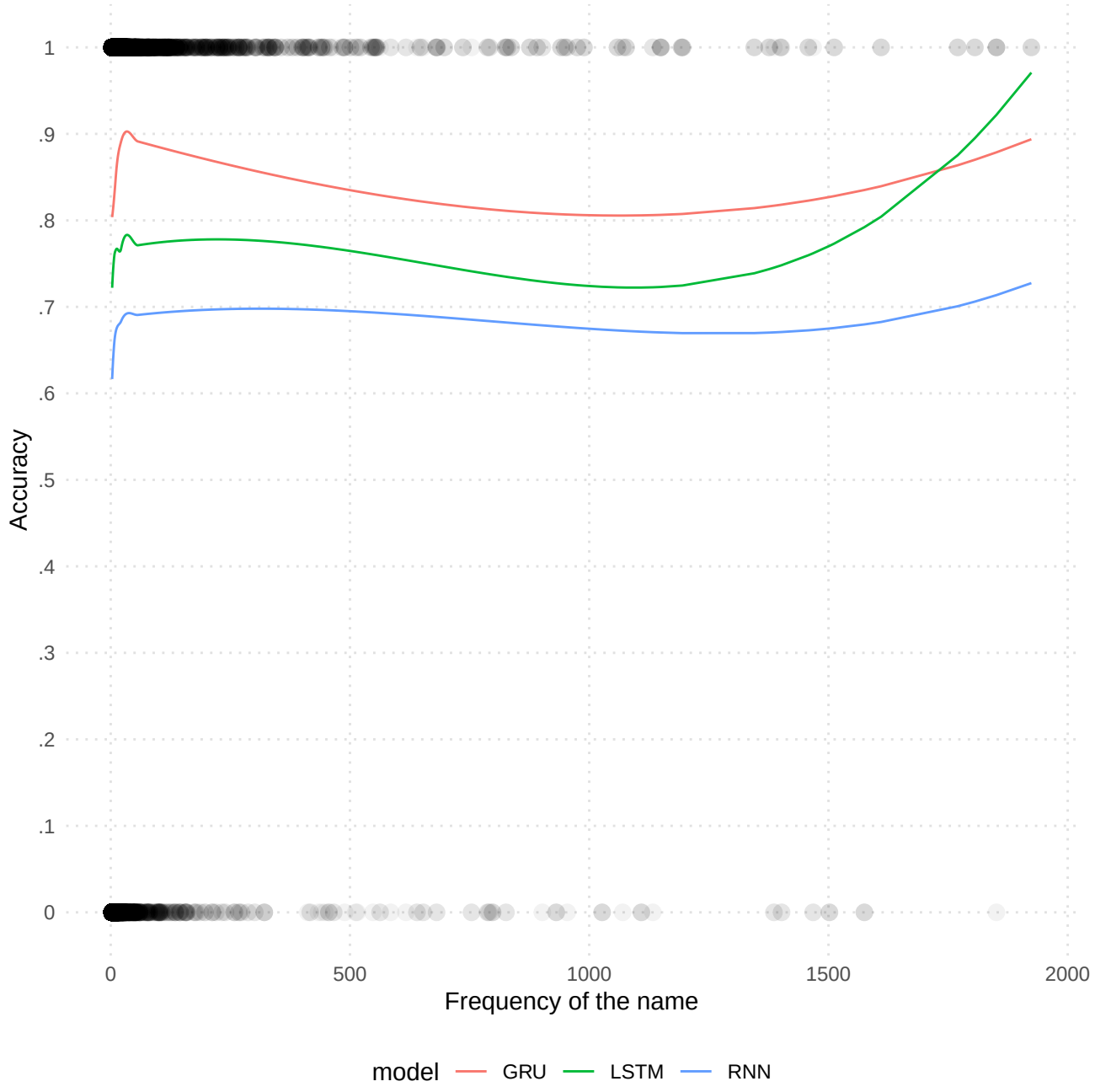
## 5 States to Languages

To create our states to languages database, we consulted the state constitutions for the official languages and the latest Indian census (the 2011 census) for the most widely spoken languages in the state. We then append the mapping to our state predictions. Our method has some weaknesses. While the modern Indian state boundaries and state education curricula are affected by language conflicts (for instance, there were anti-Hindi agitations in Tamil Nadu), there is no unique mapping between states and languages. This is for a few reasons. For one, many Indian states are massive and there is considerable linguistic diversity across regions within states. Second, internal migration, especially to urban centers, has transformed many cities. For instance, say that all Dhingras only speak Punjabi and say that all of them once upon a time lived in Punjab. Suppose that over time half of them immigrated to Delhi and never learned another language. Say that we map Delhi to people who understand Hindi. Our classifier will then indicate that Dhingras have a 50% chance of speaking Punjabi and a 50% chance of speaking Hindi. But in fact, no Dhingra can understand Hindi. Our prediction in such a case would be highly erroneous. We have reasons to think, however, that the net error is generally likely lower. For instance, in the above example, we make the slightly untenable assumption that immigrants never learn to understand the dominant language of the geography they immigrate to. This can be

**Table 2:** Accuracy of Different Models By State of Residence

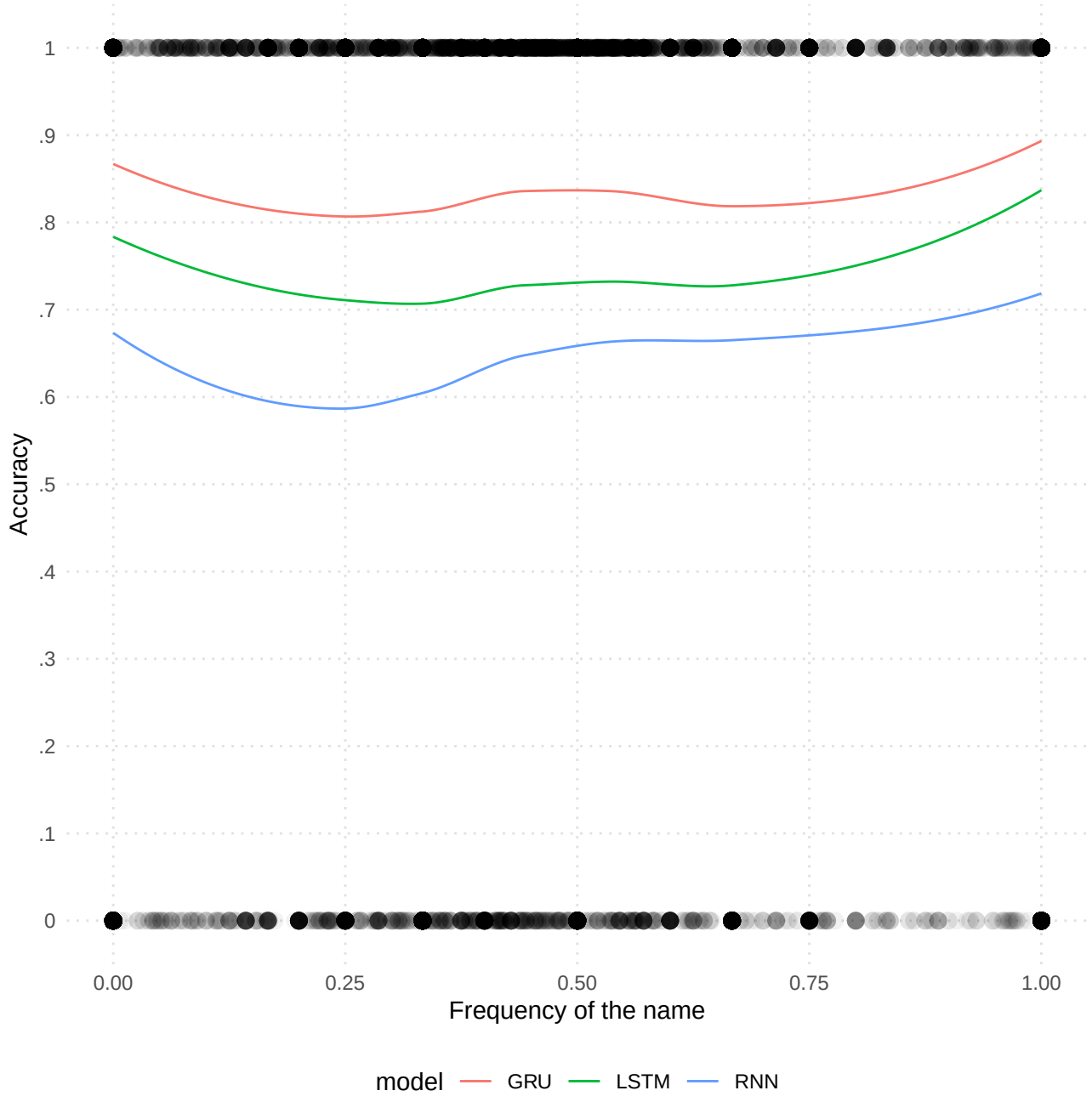
| state | RNN | LSTM | GRU |
|---|---|---|---|
| Andaman and Nicobar | 50.2 | 69.2 | 66.9 |
| Andhra Pradesh | 65.2 | 70.1 | 80.4 |
| Arunachal Pradesh | 60.0 | 81.7 | 81.2 |
| Assam | 73.4 | 93.6 | 89.0 |
| Bihar | 25.7 | 36.5 | 90.2 |
| Chandigarh | 18.6 | 23.3 | 88.4 |
| Dadra | 69.7 | 76.9 | 79.7 |
| Daman | 44.8 | 50.9 | 61.1 |
| Delhi | 22.4 | 36.2 | 37.6 |
| Goa | 34.5 | 48.1 | 53.5 |
| Gujarat | 88.1 | 91.6 | 94.7 |
| Haryana | 18.2 | 18.7 | 94.3 |
| Jharkhand | 32.2 | 45.3 | 79.1 |
| Jammu and Kashmir | 68.8 | 84.9 | 89.1 |
| Karnataka | 88.5 | 89.8 | 94.4 |
| Kerala | 26.5 | 52.4 | 50.7 |
| Maharashtra | 50.0 | 67.7 | 72.3 |
| Manipur | 31.0 | 49.8 | 54.3 |
| Meghalaya | 27.8 | 93.6 | 87.7 |
| Mizoram | 79.7 | 85.4 | 86.0 |
| Madhya Pradesh | 23.5 | 29.1 | 75.9 |
| Nagaland | 61.1 | 77.7 | 82.7 |
| Odisha | 76.0 | 91.0 | 89.4 |
| Puducherry | 36.2 | 37.5 | 54.3 |
| Punjab | 13.3 | 16.3 | 97.1 |
| Rajasthan | 14.3 | 15.6 | 81.9 |
| Sikkim | 71.3 | 96.3 | 90.2 |
| Telengana | 98.0 | 96.5 | 96.8 |
| Tripura | 88.3 | 99.2 | 97.0 |
| Uttar Pradesh | 12.6 | 18.0 | 86.6 |
| Uttaranchal | 16.6 | 21.2 | 80.1 |

**Figure 1:** Relationship Between Popularity and Accuracy Across Various Deep Learning Models




true but it is also likely that there is both selection bias—Dhingras who are open to learning new languages or may already know another language are likelier to immigrate to

**Figure 2:** Relationship Between Gender Ratio and Accuracy Across Various Deep Learning Models




Delhi—and learning—Dhingras who immigrate to Delhi learn a new language over time. The other compensating virtue of our dataset, as we noted above, is that electoral rolls still

map immigrants to their home districts. To the extent that immigrants do not forget their mother tongue, we will have good predictions.

## 6 Discussion

Our paper contributes to technologies that improve localization. However, our method of predicting languages understood by the user from their last name has a few limitations. The first is that our geographical units are very crude. Indian states are often as big as countries. For instance, Uttar Pradesh has more than 230 million people. There is also a large linguistic diversity within the states. In future versions, we hope to exploit the more granular data provided in the electoral rolls. Second, the relationship between geography and language is weaker because of internal migration. This is especially true in urban areas. Though as we note, our unique database of electoral rolls has the virtue of mapping people to their home districts. The third challenge is transliteration. There is no one standard way of transliterating Indian languages into English. We use one transliteration when the way to improve coverage may be to use k-best transliterations with the transliterations produced roughly in the frequency of how common they are. The fourth challenge is establishing last names. Our method of establishing the last name is crude. One way to improve the heuristic is to only use last words in names that are common across household members' names.

## References


BLEVINS, C. AND L. MULLEN (2015): "Jane, John... Leslie? A Historical Method for Algorithmic Gender Prediction." *DHQ: Digital Humanities Quarterly*, 9.

CHINTALAPATI, R. AND G. SOOD (2022): "Indicate: Transliterate Indic Languages to English," https://github.com/in-rolls/indicate.

CHO, K., B. VAN MERRIËNBOER, D. BAHDANAU, AND Y. BENGIO (2014): "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*.

HOCHREITER, S. AND J. SCHMIDHUBER (1997): "Long short-term memory," *Neural computation*, 9, 1735–1780.

HU, Y., C. HU, T. TRAN, T. KASTURI, E. JOSEPH, AND M. GILLINGHAM (2021): "What's in a name?–gender classification of names with character-based machine learning models," *Data Mining and Knowledge Discovery*, 35, 1537–1563.

HUA, Q., S. QUNDONG, J. DINGCHAO, G. LEI, Z. YANPENG, AND L. PENGKANG (2018): "A Character-Level Method for Text Classification," in *2018 2nd IEEE Advanced Information Management,Communicates,Electronic and Automation Control Conference (IMCEC)*, 402–406.

HUR, Y. (2022): "Malaysian Name-based Ethnicity Classification using LSTM," *KSII Transactions on Internet and Information Systems (TIIS)*, 16, 3855–3867.

IMAI, K. AND K. KHANNA (2016): "Improving ecological inference by predicting individual ethnicity from voter registration records," *Political Analysis*, 24, 263–272.

PARASURAMA, P. (2021): "raceBERT–A Transformer-based Model for Predicting Race from Names," *arXiv preprint arXiv:2112.03807*.

SOOD, G. AND A. DHINGRA (2023): "Indian Electoral Roll PDF Corpus," https://doi.org/10.7910/DVN/OG47IV.

SOOD, G. AND S. LAOHAPRAPANON (2018): "Predicting race and ethnicity from the sequence of characters in a name," *arXiv preprint arXiv:1805.02109*.

SOOD, G., S. LAOHAPRAPANON, M. SANJEEVI, AND K. TRAN (2018): "Parsed Indian Electoral Rolls," https://doi.org/10.7910/DVN/MUEGDT.

WONG, K. O., O. R. ZAÏANE, F. G. DAVIS, AND Y. YASUI (2020): "A machine learning approach to predict ethnicity using personal name and census location in Canada," *Plos one*, 15, e0241239.

WU, X. AND J. HE (2020): "Character-Level Recurrent Neural Network for Text Classification Applied to Large Scale Chinese News Corpus," in *2020 The 3rd International Conference on Machine Learning and Machine Intelligence*, New York, NY, USA: Association for Computing Machinery, MLMI '20, 83–87.

YE, J., S. HAN, Y. HU, B. COSKUN, M. LIU, H. QIN, AND S. SKIENA (2017): "Nationality classification using name embeddings," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 1897–1906.

YE, J. AND S. SKIENA (2019): "The Secret Lives of Names? Name Embeddings from Social Media," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA: Association for Computing Machinery, KDD '19, 3000–3008.