Instate: Predicting the State of Residence From Last Name [*]

Atul Dhingra[†] Gaurav Sood[‡]

March 4, 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 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

[*]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.

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 81.3% 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 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 three columns: first name, last name, and state. We discard all names that have just one word as we cannot establish the last name for these names. We assume the last word is the last name. We discard cases where the last word is less than two letters as we don't believe it is the last name. We further remove cases where the last name is not found in the father's or husband's name. 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 are present less than thrice across the dataset. Finally, we convert all the last names into lowercase.

3 Models

3.1 Naive Bayes

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

3.2 KNN

Our second model is a KNN model, in particular 1NN. We split the data into a training set (80%) and test set (20%). Using the training set as our lookup corpus, we find the closest example (in edit distance space) in the training data. Our test set top-3 accuracy—chances that our top 3 predictions have the modal state from which people with that last name live—is 44.8%. 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.

3.3 RNN

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.) The KNN model exploits 'data leakage' in that there is often a close analog to the 'unseen names' in the test set. This makes the model less useful for names that differ more dramatically. To overcome these limitations we need a model that captures the patterns in names. To that end, we build a character-level RNN model that takes the sequence of characters to predict the top three states with the highest conditional probability of people with the last name (Wu and He, 2020, Hua et al., 2018). In particular, we leverage a 2-layer RNN with 512 hidden units, train the model for 5000 epochs 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. Figure 1 graphs the Negative Loss-Likelihood loss over the epochs.

3.4 LSTM

While training the RNN we observe that the training loss is slow to converge, and plateaus after 5000 epochs, to improve the model capacity and avoid the major RNN pitfalls, we

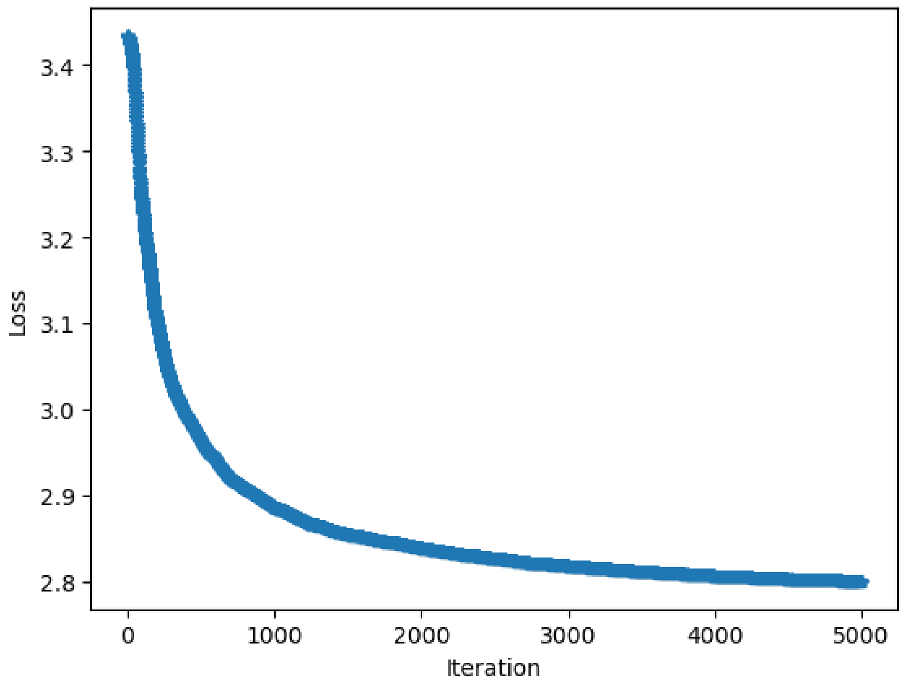

Figure 1: Negative Log Loss with RNN

build a LSTM model that takes the sequence of characters to predict the top three states with the highest conditional probability of people with the last name. We build a 2-layer LSTM with 512 hidden units, train the model for 10000 epochs with a batch size of 256 using Adam optimizer, and use a negative log-likelihood loss. We set a low learning rate of 3e-4. Figure 2 graphs the Negative Loss-Likelihood loss over the epochs.

3.5 GRU

One of the major drawbacks of using a 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 designed a character-level GRU model that takes the sequence of characters to predict the top three states with the highest conditional probability of people with the last name. we designed a 2-layer GRU 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. Figure 3 graphs the Negative Loss-Likelihood loss over the epochs.

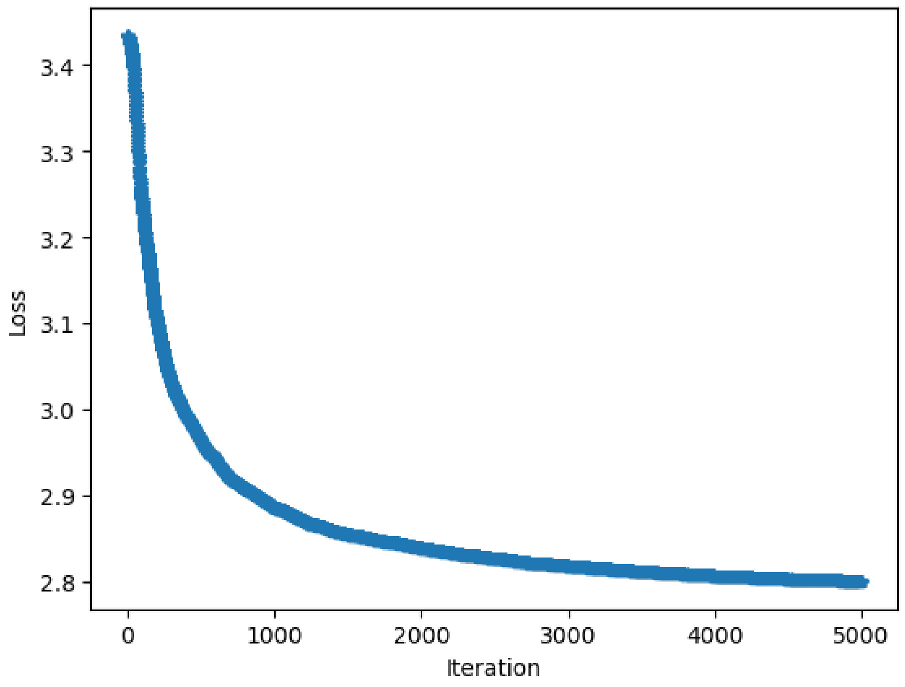

Figure 2: Negative Log Loss with LSTM

Apart from complete test-set evaluation, we also analyzed the performance of the various models with randomly sample k(k=3000) unique names.The baseline with 1NN with edit distance model predicts 58% correctly, as compared to 95.9% with our best performing GRU model. However, in real life the type of inference requests coming in depend on how frequent a name occurs, for this reason, we provide an in-depth comparison of our models across various weighting strategies, as shown in table-1

We split the data by last names, which keeps the last names distinct between the two sets. We keep 80% of the data for training and 20% for testing. On the test set, the top-3 accuracy—-chances that our three predictions include the modal state in which people with that last name live—of our model is 81.3%. Compare this to the no-information baseline where we only know the population of the states and guess the modal state. The most populous state in India is Uttar Pradesh and if we were to guess that as the answer for all the names, we would be right less than 16.74% of the time. The 1NN with edit distance results can be seen as another baseline but as we write above, we believe the 1NN results reflect the fact that there is a quasi-overlap between training and test sets. Technically, the overlap should help with the RNN though our RNN is shallow and cannot learn very complex patterns.

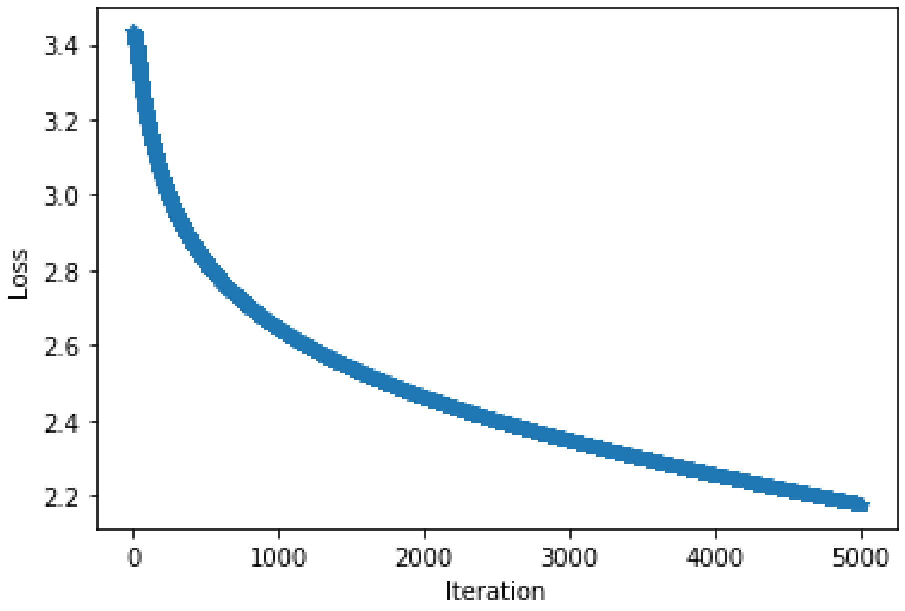

Figure 3: Negative Log Loss with GRU

4 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 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

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.

5 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 that is provided in the electoral rolls. Second, because of internal migration, the relationship between geography and language is weaker. 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 transliterations. 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.

6 Appendix

References

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

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.

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

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

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.