Data Mining and Knowledge Discovery
https://doi.org/10.1007/s10618-021-00748-6

What's in a name? – gender classification of names with character based machine learning models

Yifan Hu[1] · Changwei Hu[1] · Thanh Tran[2] · Tejaswi Kasturi[1] · Elizabeth Joseph[3] · Matt Gillingham[3]

Received: 5 November 2019 / Accepted: 5 March 2021

Abstract
Gender information is no longer a mandatory input when registering for an account at many leading Internet companies. However, prediction of demographic information such as gender and age remains an important task, especially in intervention of unintentional gender/age bias in recommender systems. Therefore it is necessary to infer the gender of those users who did not to provide this information during registration. We consider the problem of predicting the gender of registered users based on their declared name. By analyzing the first names of 100M+ users, we found that genders can be very effectively classified using the composition of the name strings. We propose a number of character based machine learning models, and demonstrate that our models are able to infer the gender of users with much higher accuracy than baseline models. Moreover, we show that using the last names in addition to the first names improves classification performance further.

Responsible editor: Ira Assent, Carlotta Domeniconi, Aristides Gionis, Eyke Hüllermeier

✉ Yifan Hu
yifanhu@verizonmedia.com

Changwei Hu
changweih@verizonmedia.com

Thanh Tran
tdtran@wpi.edu

Tejaswi Kasturi
kasturit@verizonmedia.com

Elizabeth Joseph
ejoseph@verizonmedia.com

Matt Gillingham
mgilling@yahoo.com

[1] Yahoo! Research, Verizon Media, 770 Broadway, New York, USA

[2] Worcester Polytechnic Institute, Worcester, MA, USA

[3] Verizon Media, 701 1st Ave, Sunnyvale, CA, USA

Published online: 12 May 2021

Keywords Natural language processing · Neural network · Character-based machine learning model · Demography · Gender

1 Introduction

Gender information is no longer a required input when registering for an account at many leading Internet companies. Consequently, a significant percentage of registered users do not have a declared gender; and over time this percentage is expected to rise. In many tasks, such as when checking for fairness in machine learning algorithms for content recommendation (Karako and Manggala 2018; Yao and Huang 2017), it is important to know users' demographic information, of which gender is an integral part. Therefore, it is necessary to infer the gender of those registered users who do not disclose it, by using information collected during their registration, and/or from their subsequent activities. Although we are aware that gender identity is not necessarily binary, we treat gender inference as a binary classification problem in this paper. In addition, this research was conducted in accordance with Verizon Media's Privacy Policy and respect for applicable user privacy controls.

One possible way to infer gender is to examine user activities. We experimented with a content based model, where features are extracted from the content the users interacted with through activities such as browsing. The content is distilled into categorical features including wiki-entities (i.e., entries in Wikipedia), and proprietary content categories derived via ML models similar to Google Cloud Content Categories (2019). These categorical features are used in a logistic regression algorithm to classify user genders. The advantage of such a content-based model is that it identifies users by their browser's cookies, thus it works even for users who are unregistered or not logged in. However, a limitation of such content-based approaches is that signals for users can be sparse. Additionally, signals may be lost in the process of classifying content into categorical features. As a result, we found that the performance of this content-based approach is modest.

Therefore we are interested in enhancing the above baseline model for registered users, for whom we have more information. In particular, we believe that the user's name could be used to infer gender effectively. Accordingly, we propose a number of character-based machine learning approaches for the gender classification task. Our experimental results on 100M+ users demonstrate that the name-based approach improves AUC (of ROC) by 0.14 compared with the content-based model.

The major contributions of our work are:

- We propose several character-based techniques to predict users' genders based on their first name. By constructing both a linear model and various deep neural models, we show that gender can be predicted very effectively using information encoded in the spelling of the user's first name.
- We find surprisingly that a linear model, combined with judicious feature engineering, performed just as well as complex models such as character-based LSTM (Hochreiter and Schmidhuber 1997) or BERT (Devlin et al. 2018), and bet-

ter than other character-based deep learning models, an embedding-based model, and a baseline content-based model.
- We conduct our experiment on a very large real-world dataset of 21M unique first names and show that our models trained on this large dataset extend better than the same models trained on a medium sized public dataset. We also show that our models perform well on unseen or multilingual datasets.
- We demonstrate that the performance could be further enhanced when utilizing both the first and the last names through our proposed dual-LSTM algorithm. We show that fusing name and content information can also boost performance.

Throughout the paper, we shall denote $P(M)/P(F)$ the probability of male/female, and $P(M|X)$ the probability of male given feature set X. The goal of the proposed machine learning models is to compute $P(M|X)$, based on the user feature set X. All models in this paper minimize the binary entropy loss function.

The outline of this paper is as follows. Section 2 introduces datasets used for our models. Section 3 proposes numerous character-based models for gender classification. Section 4 compares performance of these models. Section 5 utilizes both the first and last names to enhance performance. Section 6 reviews related work. We conclude in Sect. 7 with discussions on limitations and future work. In the Appendix, we propose a fusion model utilizing both name and content, and discuss details of using the gender prediction models in practice, as well as how to understand the models.

2 Data and the baseline model

To infer genders of registered users, at least two kinds of data can be used. One is the content that the users interacted with, and the other is the declared information of the users during registration, particularly names. The former is the premise for a content-based baseline model we tested. However we believe that the latter could be an important signal for inferring the gender. Our work is based on two datasets, one from the Social Security Administration (SSA), and the other from Verizon Media. Throughout this paper, when we talk about the name of a person, we are referring to the first name, unless otherwise specified.

2.1 SSA baby names dataset

The Social Security Administration collects data of names, genders, and name frequencies (Social Security Administration 2018) for newborn babies in the US, spanning from 1880 to 2018 (139 years). For babies born in 2018, the top 4 most popular names are "Olivia (female, frequency: 17,921); Noah (male, frequency: 18,267), Emma (female, frequency: 18,688), and Liam (male,frequency: 19,837)". Around 11% of the names appear both as male and female. For example, in 2018, the name "Avery" is documented as both female (8053 times) and male (2098 times). Henceforth, we will call this dataset SSA Data. When aggregating over all 139 years, we found 98,400 unique names from a total of 351,653,025 babies, of which 50.5% are boys. Among

Table 1 Left: top male and female US baby names in SSA DATA. Right: top male and female names in YAHOO DATA. Here $p(M)$ is the observed probability of the name being male

SSA DATA				YAHOO DATA			
Male name	Count	Female name	Count	Male name	Count/$p(M)$	Female name	Count/$p(M)$
James	5.2M	Mary	4.1M	John	2.4M/0.977	Maria	1.0M/0.031
John	5.1M	Elizabeth	1.6M	David	2.1M/0.981	Mary	0.9M/0.054
Robert	4.2M	Patricia	1.6M	Michael	1.9M/0.983	Jennifer	0.9M/0.027
Michael	4.4M	Jennifer	1.5M	James	1.7M/0.974	Jessica	0.8M/0.021
William	4.1M	Linda	1.5M	Robert	1.4M/0.979	Sarah	0.8M/0.025

Table 2 Percentage of users in YAHOO DATA with name frequency $\geq k$

k	Name count	Female coverage	Male coverage
5	3.1M	92.65	92.43
10	1.7M	90.84	90.60
20	1.0M	88.96	88.61
40	0.5M	86.93	86.29
80	0.3M	84.74	83.85
100	0.3M	83.59	82.71

the 98,400 unique names, 10,773 of them are used by both boys and girls. Table 1 (left) shows the top 5 male and female names.

2.2 Verizon media dataset

This dataset is a sampled dataset from Verizon Media (formerly Yahoo! and AOL), which contains a total of 100M+ of users. To be consistent with SSA DATA, we aggregate by first name to give us the final dataset, which contains "first name, gender, #male, #female." Here "first name" can contain one or more tokens, such as "John Robert". We assign a label of 1 if there are more men using this first name than women, and 0 otherwise (please see Sect. 4 for a probabilistic treatment of labels). This dataset, called YAHOO DATA from now on, contains 21M unique first names. They follow a typical power law distribution with a long tail. Over 72% of names only appear once, 10% appear twice, and 4% appear 3 times. On the other end of the scale, the most popular names are shown in Table 1 (right). Compared with SSA DATA, the popular names are different. This reflects both the fact that Verizon Media users are international (for example, "Maria" is a popular female name in Spanish speaking countries), and the fact that the popularity of names changes over time and that the SSA DATA covers 139 years. In addition, YAHOO DATA is less certain compared with SSA DATA. For example in Table 1, top names are around 97.0% male or female, while these same names are recorded as 100% male or female in SSA DATA.

Table 2 shows the ratios of users that are covered by "popular" names. If we take all 0.3M names that appear at least 100 times in the data, 84% of men and 83% of women are covered. If we take all the 3.1M names that appears at least 5 times, 92% of users are covered.

Note that the size of the set of "popular" names in YAHOO DATA is considerably larger than the whole SSA DATA. Even imposing a high frequency of 100 or more, we end up with 300K names, *vs.* 98K names in the SSA DATA. One reason for the larger size of the set of popular names is that the names in YAHOO DATA may contain multiple tokens. About 34% of names contain a mixture of letters and spaces, e.g. "Ana Carolina" or "Carlos Eduardo". Another reason is that Verizon Media has a substantial international presence. A check on a random sample of YAHOO DATA shows that about 2.2% of the names contain non-English characters. The most popular of such names include "André" and "Björn" for men, and "Aurélie" and "Bárbara" for women. Moreover, around 11% of first names contain non-letters, such as "Anne_Marie", "Asma'a".

Since we are interested in inferring gender for Verizon Media users, and given the significant differences between the SSA DATA and YAHOO DATA, we propose to train a number of name-based machine learning models using YAHOO DATA, but shall verify our models using both SSA DATA and YAHOO DATA.

Finally, in order to study the effect of using full names (Sect. 5), we randomly sampled 13M users, each with a full name, together with the declared genders, out of the 100M+ users in the YAHOO DATA. We call this set YAHOO FULL NAME DATA.

2.3 Content data for the baseline model

The baseline content-based model uses around 5.5 million features extracted from content that the users interacted with. These user activities include page views and clicks. Each of these activities is converted to a category represented by a wiki-entity[1] or an internal Yahoo category. Then, a logistic regression model is applied on these features to predict gender of an input user.

3 Models for gender prediction

There are multiple possible approaches to infer a user's gender. A content-based baseline model uses categorical features extracted from content the user interacted with. It has the advantage of identifying users by browser cookies, so works for users who are not registered or not logged in. Its primary disadvantages are that content signals could be sparse, and signals may be lost in the process of distilling into categorical features. Consequently, the performance is modest (AUC of ROC around 0.80).

Since the goal of this paper is to enhance the gender prediction accuracy for registered users who did not declare their genders, we propose to use their first names to infer their genders. We note that while the first name might be a rather random combination of letters, there's still a societal mechanism that makes us interpret certain

[1] E.g., if a user read an article about the department store Macy's, a categorical variable `wiki_Macy's` is added to the list of features describing the user

names as female or male. This means that ad-hoc creation of a name won't work but deciphering this societal gender coding via machine learning can work.

Given the first name of a user, there are two approaches to infer the gender. One approach is to use the name embeddings as features. In a prior work, Han et al. (2017) established ethnic and gender homophily in email correspondence patterns, namely that on aggregate, there is greater than expected concentration of names of the same gender and ethnicity in the contact lists of email users. Consequently they constructed name embeddings by applying word2vec (Mikolov et al. 2013) to the contact lists of millions of users, and used the embeddings to infer the nationality and cultural origins (Ye et al. 2017). We implemented this approach as a second baseline. However, while the embedding based approach works well for most users, word2vec requires a name to appear enough times in contact lists to be able to build meaningful embeddings. For languages such as English, there exists a set of given names that most people adopt. Nevertheless, there are variations of common first names that are less often seen. For example, "Liliana" is in SSA DATA, but its variation "Lilianna" is not in SSA DATA. This problem of rare names is far worse for some character-based languages such as Chinese or Japanese, where a person's first name consists of a few characters in a set of tens of thousands of characters. Thus, first names in these languages are of a very high cardinality and are often unique to a person. Consequently, we found that the embedding-based approach misses many names in its vocabulary set, leading to a degraded performance.

3.1 Why character-based models?

Instead of using name embeddings, we propose character-based approaches to infer the gender, because certain characters, or a combination of a few of them, convey the gender of a person. For example, in English, first names ending with "a" ("Linda", "Maria" etc) are more likely female names. Figure 1 (left) shows a distribution plot of the last character in names. We can see that 34.7% of women have first names that end with "a", vs 1% of men. In Chinese, certain characters are only used for female (e.g., 丽, meaning "beautiful"), or for male (e.g., 勇, meaning "brave").

Beyond single characters, character n-grams can also help to infer genders. For example, Figure 1 (right) shows a heatmap of $P(F)$ (female probability) based on the last two characters of names. For example, names end with "?a" (last row) are mostly females, except for "ua", which has a 98.1% probability of being male (e.g., "Joshua"). In fact, names where the second last character is "u" (6th last column) mostly refer to males. Overall, out of 326,765,644 people in SSA DATA, a large percentage (78.7%) of them have names that end with a character 2-gram that can be found in Fig. 1 (right), which implies $P(F) \geq 0.7$, or $P(M) \geq 0.7$.

Therefore, we believe that character n-grams are correlated with gender. This leads us to build character-based models trained from millions of names and their known genders. *A key advantage of character-based approaches is that since the features used are characters and character n-grams, the models can work on all names, regardless of whether they are popular or rare.* In particular, it does not suffer from the out-of-vocabulary words issue that the embedding-based approach faces.

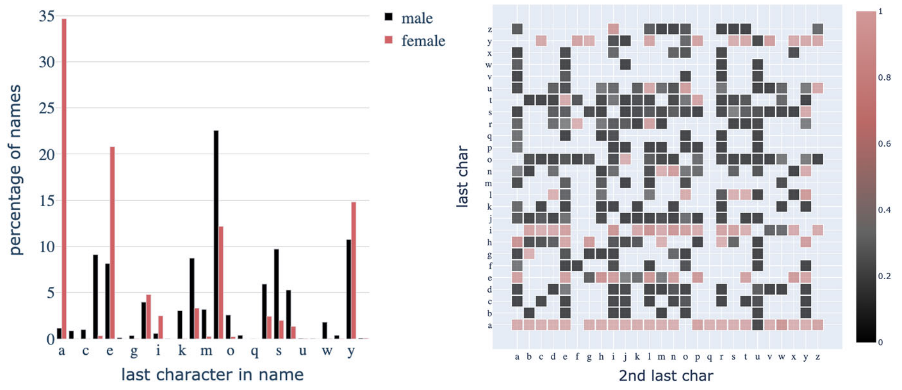


Fig. 1 Left: distribution plot of the last character in SSA DATA among boys and girls. Best viewed in color. Right: heatmap of the female probability, $P(F)$, given the last two characters in SSA DATA. X-axis is the 2nd last character, y-axis the last character. Only showing character 2-grams with $P(F) > 0.7$ or $P(F) < 0.3$, and frequency ≥ 40

In the following, we present a number of character-based models for the gender classification task. We will discuss these models at the algorithmic and architectural level. Implementation details and results will be presented in Sect. 4.

3.2 Class scaled logistic regression (NBLR)

An effective yet simple model often used in text classification is NBSVM (Wang and Manning 2012). In this paper we used a similar model, except that we replace the SVM with logistic regression. The resulting algorithm is referred to as class scaled logistic regression, but is denoted as NBLR in contrast to NBSVM. We describe its details in Algorithm 1. The input includes first names and declared gender labels. The model leverages character n-grams, tf-idf and log ratio information for the prediction of gender.

Algorithm 1 Class Scaled Logistic Regression (NBLR)

```
• Form character n-grams of the name corpus (we use $n$ between 1 to 7).
• Compute the tf-idf (sparse) matrix $X$.
• Compute the log ratio $r_j$ for each column $j$ of $X$, then scale the column with $r_j$.
• Train a logistic regression model using the scaled matrix $X$ and the labels.
```

The log ratio r_j of feature j measures the log-odds of the feature. Specifically, let the feature matrix be $X \in R^{m \times n}$, binary labels be $y \in \{0, 1\}^m$, then r_j is defined as the logarithm of the ratio between the average value of the elements of column j of X associated with positive labels in y, and the average value associated with negative labels: $r_j = \log\left(\frac{(1+\sum_{i:y_i=1} X_{ij})/(1+\sum_{i:y_i=1} 1)}{(1+\sum_{i:y_i=0} X_{ij})/(1+\sum_{i:y_i=0} 1)}\right)$.

3.3 DNN

A fully connected deep neural network (DNN) is a versatile nonlinear model that, given a large set of training data, can learn the weights of hidden layers to minimize the final loss function. We used a DNN with multiple hidden layers. As input to the DNN, we first encode each character in a name with a one-hot vector, and then concatenated one-hot vectors associated with each character to a longer binary vector based on the order of their occurrences in the name.

3.4 Byte-CNN

Character-level CNN (char-CNN) has also been applied to natural language processing (Zhang et al. 2015). In char-CNN, each character is represented by a vector of one-hot encoding or fixed-length embedding. These vectors are concatenated into a matrix to represent a sequence of characters (one or multiple sentences). One dimensional CNN is applied to this input matrix. In doing so, CNN combines nearby characters and essentially builds up a hierarchy of character grams. Character level encoding is useful for English. But for multilingual text and text with complex symbols like emojis, character level encoding will introduce a lot more parameters. Since our YAHOO DATA is multilingual and contain many emojis, we propose byte-CNN, a modified version of char-CNN which uses UTF-8 encoding to reduce the number of parameters.

3.5 LSTM

Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) is a recurrent neural network based model that is especially suited for sequence data, in this case characters. In the gender classification task using input names, we consider each character of an input name as a unit and parameterize each character as an embedding vector. Then, we model the transition from characters to characters. Together with a traditional LSTM design, where the output of the last character is used for classification tasks with an intuition that it summarized all information from previous characters, we consider a pooling layer (i.e. Fig. 2 (left)) to combine results from all outputs of LSTM cells. In this paper, we experiment with *max* and *mean* pooling.

3.6 Char-BERT

This model originated from a recent state-of-the-art language model BERT (Devlin et al. 2018). Instead of using tokens from WordPiece (Wu et al. 2016), here the basic units are characters. Figure 2 (right) shows the character-based BERT architecture for the gender prediction task with an input name “John”. Similar to the LSTM model, we encode each character of the input name by an embedding vector. Note that *[CLS]* and *[SEP]* tokens are inserted into the beginning and ending of the input name, analogous to the BERT design in Devlin et al. (2018). *[CLS]* and *[SEP]* are also encoded by two

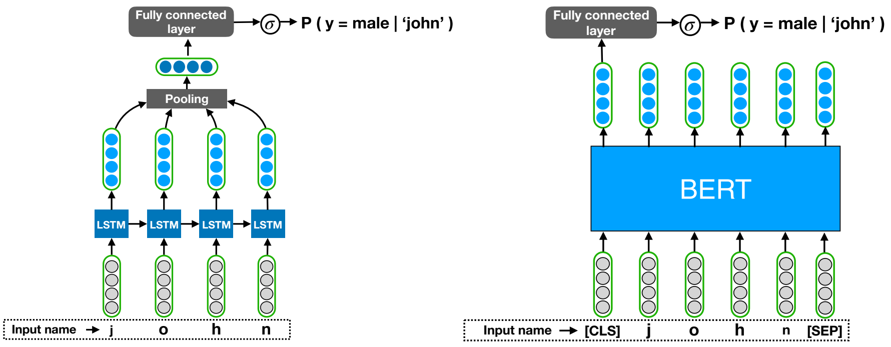


Fig. 2 Left: character-based LSTM model with pooling. Right: character-based BERT model

separated embedding vectors. Next, the output embedding vector of the *[CLS]* token is used for the gender classification task as it self-attentively aggregates representations of all other tokens in the input name.

3.7 Name embedding (EMB)

This is a baseline model using name embeddings (Han et al. 2017; Ye et al. 2017). Embeddings of first and last names are built by the word2vec (Wu et al. 2016) model using the contact lists of millions of users as "sentences". They are then used as feature vectors in training and testing.

4 Experimental results

We consider the gender prediction problem as a binary classification problem. Given that some names are used by both men and women, it may appear at first that this should be considered as a regression problem[2]. Even though 10,773 out of 98,440 (11%) names in SSA DATA are unisex, our preliminary tests on some of our models show that treating gender prediction as a regression problem does not perform as well as treating it as a binary labelled classification problem. Hence, for the rest of this paper, we consider gender prediction as a binary classification problem.

The simplest way to label a name with a binary label is to use majority voting. This is a reasonable strategy especially considering that the YAHOO DATA clearly contains noise. For example, names such as "John" and "Maria" that are of single-gender in SSA DATA are not observed to be so in YAHOO DATA (Table 1). Thus, using the majority voting to decide the gender is helpful in these unambiguous cases. A more principled way is to treat the problem as one of *probabilistic binary classification*. For example, "Avery" is documented as male 2098 times and female 8053 times in the SSA DATA, hence we can have two samples of "Avery", one with the label of 1 and a sample weight

[2] As an example, in SSA DATA, around 21% of people with the name "Avery" are male. In a regression setting, we can fit a model to predict a value of 0.21 given this name. On the other hand, in a binary classification setting, we seek to predict a value of 0 for this name.

of 2098, the other with the label of 0 and a sample weight of 8053. We experimented with this probabilistic treatment in combination with NBLR. We call the resulting algorithm CSLR$_w$. We found later in Table 4 that CSLR$_w$has similar performance compared with NBLR, indicating that treating the gender classification as a pure binary classification problem, as opposed to a probabilistic binary classification problem, is sufficient, with the advantage of reduced memory and computational requirements.

We split the YAHOO DATA containing 21M unique names randomly into train, validation and test sets with a ratio of 70/10/20, respectively. We train our models using the train set, fine tune model parameters with the validation set, and report results on the YAHOO DATA test set, and on the SSA DATA. Notice that because the size of the YAHOO DATA is quite large, the variability when using different random splits is small. For example, when using 5-fold validation on the 21M YAHOO DATA, the standard deviation of all versions of AUC and accuracy (see Sect. 4.1) are below 0.005. Therefore in the following we only report results on the above randomly split data sets.

4.1 Performance measurement

Even though we build binary classifiers, when measuring performance, each name should not be treated equally. For example, it is more important to predict the genders of popular names correctly, because errors on these names affect more people. Thus, we use two versions of metrics. The first version treats *each name* as a sample. For example, SSA DATA has 98,400 unique first names, thus 98,400 samples. We denote the AUC of ROC and accuracy for this version as AUC$_0$ and ACC$_0$.

Since popular names represent more people, in real applications it is often more important to predict the gender correctly for these names. Therefore the second version of the metrics treats *each user* as a sample. For example, in SSA DATA, ‘Avery’ was used by 54,330 boys and 125,747 girls. Thus when computing metrics, we split ‘Avery’ into two instances (male vs female) that are assigned the same probability $P(M|Avery)$, the male one has a sample weight of 54,330, and female one has a sample weight of 125,747. We denote the resulting AUC of ROC and accuracy as AUC and ACC.

4.2 ML models and features

We experimented with six character-based ML algorithms. In the following we discuss each of these algorithms and its parameter setting. All our character based models take into account of the fact that the sequential ordering of the characters in a name is important, and encode that automatically. The exception is the NBLR algorithm, which has to take the sequential ordering into account both implicitly in the character n-gram formation, and also explicitly (see below). For Byte-CNN, DNN, LSTM and char-BERT, we limit the max number of characters to 30, padding empty spaces in front if the name is less than 30 characters, or dropping the part of the name beyond 30 characters.

NBLR: this is the class scaled logistic regression with L_1 regularization. For features, we used character $1, 2, \dots, n$-grams. We experimented with different values of n, and found that the performance improves as n increases, but the improvement slows when $n \geq 3$. The performance stabilizes when $n \geq 6$. We thus set $n = 7$ for the rest of the experiments. Importantly, we found that if we preprocess the name token by prefixing and suffixing each token with "_", the AUC can be improved by around 1-3%. This is because without this preprocessing, a bigram such as "ia" could be the last two characters from "Julia", or the middle two characters from "Brian". There is no way to differentiate these two cases. With the preprocessing, we know that a 3-gram "ia_" means that the name ends with "ia". Incidentally, we realized afterwards that this is similar to how WordPiece (Wu et al. 2016) tokenizes text. The difference is that WordPiece adds a "__" to the beginning of a word, but not to the end of a word.

Byte-CNN: To handle multilingual names and emojis, we proposed byte-CNN in which we encoded multilingual strings or emojis using UTF-8. UTF-8 is a superset of ASCII, and used one byte to encode the ASCII set for the first 128 characters, which include all letters, numbers and regular symbols. For contents that can not be encoded with ASCII codes, UTF-8 uses more than one byte for encoding. For instance, UTF-8 encodes one Chinese character as three bytes. For SSA DATA, since all names are in English, byte-CNN is equal to char-CNN in this case. For multilingual YAHOO DATA, UTF-8 encoding results in 178 unique encoded bytes. We used an embedding layer with 512 dimensions to encode all 178 unique bytes, and then adopted a deep architecture similar to char-CNN, but the embedding is applied on byte level instead of character level. The byte-CNN network includes three dilated convolution layers with 64, 128, and 256 kernels, respectively, with a kernel size of 3 for all three layers.

DNN: Similar to byte-CNN, we used UTF-8 to encode multilingual YAHOO DATA. A one-hot vector was used to represent a UTF-8 byte, and a fixed number of these one-hot vectors were concatenated into a long vector to represent a name. We experimented with numerous configurations of a fully connected deep neural network. The optimal configuration has 5 layers with corresponding 1024, 1024, 512, 256, and 128 hidden nodes, and with a dropout rate of 0.2.

LSTM: Given an input name with l characters, an LSTM model will output l embedding vectors. As mentioned before, we processed the l output embedding vectors in 3 strategies: (i) using the last output embedding vector ; (ii) using a *max* pooling layer on top to combine ; (iii) using an *average* pooling layer on top to aggregate . We experimented with all three versions, and found that the first version with an embedding size of 192 provided the best results on the validation set. Hence, we only report results from this version.

char-BERT: *char-BERT*'s architecture is the same as that of *BERT*, and *char-BERT* can be seen as a pure attention network of only transformation operators without using any language-based pretraining. Note that we can pretrain *char-BERT* with a masked character prediction task, but it is out-of-scope in this paper and we leave it for a future work. For LSTM and char-BERT, we varied the embedding sizes from {12, 24, 48, 96, 128, 192}, fixed the learning rate at 0.001, and optimized with Adam optimizer (Kingma and Ba 2014). For *char-BERT* model, we varied the number of attention heads and layers from {2,3,4,6}. We found that *char-BERT* obtained its best results

Table 3 Performance of NBLR models trained on YAHOO DATA with name frequency $\geq \upsilon$ and tested on the whole SSA DATA

υ	AUC_0	AUC	ACC_0	ACC
1	0.958	0.991	0.897	0.963
2	0.951	0.990	0.885	0.957
5	0.940	0.986	0.874	0.944
10	0.930	0.98	0.862	0.934
20	0.921	0.972	0.853	0.920
40	0.914	0.962	0.846	0.907
80	0.906	0.945	0.837	0.882
100	0.902	0.940	0.833	0.880

on validation set when using 3 layers with 6 attention heads and an embedding size of 192. We use this setting to get the results reported in the paper.

EMB: We used the name embeddings from Han et al. (2017), which filter out names that appear less than 10 times in the contact lists. Consequently low frequency names in our training and testing data may not have an embedding, in which case we set them to a vector of zeros. An XGBoost model was trained using the embedding feature vectors and the labels. We also built a logistic regression model and found its performance to be similar. So we did not report its results here.

Note that we do not further compare with other attention-based models such as Attention-based RNN (Wang et al. 2016; Chen et al. 2017; Zhou et al. 2016) because char-BERT, which originated from BERT (Devlin et al. 2018), is a purely attentive transformer network, and has outperformed all previous models.

4.3 Low frequency names for training

We consider whether to use all training data to build models. Since labels for unpopular names may be unreliable, at first glance we should restrict training to popular names with high frequency (frequency $\geq \upsilon$). To decide the value of υ, we train a NBLR model on YAHOO DATA train set, but with name frequency $\geq \upsilon$. Since SSA DATA can be considered reliably labelled, we use the whole SSA DATA as a test set for this experiment, filtering out names that existed in the train data.

Table 3 shows the NBLR's performance when varying υ in a range of $\{1, 2, 5, 10, 20, 40, 80, 100\}$. Clearly, it improves as we lower the frequency threshold, and using all available training data gives the best test performance on SSA DATA. This indicates that even though individual labels for low frequency names may be unreliable, collectively, these names still provide valuable signals that improve classifier performance. Therefore, we proposed to use all available training data for the rest of this paper.

4.4 Comparing models on YAHOO DATA

For the rest of the paper, we denote names with name frequency $k < 40$ as unpopular names, for reasons explained in Sect. A.2. We first compare the performance of

Table 4 Top: performance for different ML models, trained with all YAHOO DATA training data, tested on testing data with name frequency < 40. Bottom: tested on all testing data

model	AUC_0	AUC	ACC_0	ACC
	on test data with name frequency $k < 40$			
NBLR	**0.879**	**0.872**	**0.799**	**0.797**
CSLR_w	0.859	0.856	0.783	0.785
DNN	0.850	0.851	0.775	0.780
Byte-CNN	0.838	0.836	0.765	0.767
LSTM	**0.879**	**0.872**	**0.799**	**0.798**
char-BERT	**0.881**	**0.874**	**0.801**	**0.799**
EMB	0.717	0.724	0.676	0.685
	All test data			
NBLR	**0.879**	**0.940**	**0.800**	**0.876**
CSLR_w	0.860	0.935	0.785	0.873
DNN	0.851	0.903	0.776	0.837
Byte-CNN	0.839	0.889	0.766	0.814
LSTM	**0.880**	**0.938**	**0.801**	**0.871**
char-BERT	**0.883**	**0.935**	**0.803**	**0.871**
EMB	0.722	0.927	0.678	0.854
CONTENT	-	0.800	-	0.729

different models on YAHOO DATA test data with these unpopular names. The results are given in Table 4 (top). EMB – the embedding-based baseline model performed poorly, because the embedding for many of these low frequency names do not exist. DNN, while being a more complicated model, actually performed worse than the linear model NBLR. The best performance comes from the NBLR, LSTM and char-BERT models, with AUC of 0.872/0.872/0.874, respectively. These three models are very similar in performance across all metrics.

The bottom of Table 4 shows the performance of the CONTENT baseline on all test data (e.g. regardless of the name frequency). CONTENT uses features extracted from the content the users interacted with, as discussed in Sect. 2.3. The AUC is 0.800, with ACC of 0.729. For comparison, NBLR has AUC of 0.940, with corresponding accuracy of 0.876, both much higher than the CONTENT baseline. LSTM and char-BERT also performed well. The other baseline model EMB performs better than the CONTENT model, with AUC on all test data of 0.920 and corresponding accuracy of 0.855, though inferior to NBLR, LSTM or char-BERT. CSLR_w, which treats the gender prediction problem as a probabilistic binary classification problem, has similar or slightly worse performance compared with NBLR.

We also checked the names on which our models do not work well. We define the loss of a prediction on a name as $(P(M|\text{first}) - 0.5)\,(|F| - |M|)$ where $|M|$ and $|F|$ are the number of men and women with that name. Names with the highest losses are those that are relatively popular, and where the prediction is wrong. The top 10 names with the highest loss for NBLR are shown in Table 5. Most of them are names commonly used by both men or women, such as "Robin". One exception is "Negar", which is

Table 5 Names with the highest prediction loss. M/(M+F) is the observed male probability

	Robin	Sina	Jan	Justine	Micah	Negar	Zia	Val	Asal	Lane
M/(M+F)	0.38	0.8	0.46	0.37	0.70	0.05	0.79	0.37	0.14	0.77
P(M)	0.74	0.16	0.94	0.72	0.25	0.72	0.17	0.84	0.81	0.28

Table 6 Testing the generalizability of the models trained on YAHOO DATA or SSA DATA

			any k		$k < 40$	
Model	Train	Test	AUC	ACC	AUC	ACC
NBLR	Yahoo	SSA	0.972	0.916	0.963	0.905
DNN	Yahoo	SSA	0.970	0.926	0.962	0.907
Byte-CNN	Yahoo	SSA	0.955	0.901	0.949	0.888
LSTM	Yahoo	SSA	**0.980**	**0.940**	**0.971**	**0.925**
Char-BERT	Yahoo	SSA	**0.980**	0.930	**0.971**	0.921
NBLR	SSA	SSA	**0.986**	**0.947**	0.977	0.939
DNN	SSA	SSA	0.978	0.938	0.951	0.897
Byte-CNN	SSA	SSA	0.971	0.926	0.940	0.877
LSTM	SSA	SSA	**0.985**	**0.953**	**0.992**	**0.960**
Char-BERT	SSA	SSA	0.955	0.891	0.982	0.936
NBLR	SSA	Yahoo	0.859	0.791	**0.756**	**0.710**
DNN	SSA	Yahoo	**0.874**	**0.823**	0.700	0.674
Byte-CNN	SSA	Yahoo	0.852	0.790	0.687	0.663
LSTM	SSA	Yahoo	0.871	0.812	0.710	0.682
Char-BERT	SSA	Yahoo	0.849	0.797	0.683	0.668

a female name of Persian origin. This indicates that inference for unisex names is a challenging task. In Sect. 5 and Sect. A.1 we offers two possible enhancement.

4.5 Generalizability of the models

In this section, we are interested in understanding whether models trained on the large scale YAHOO DATA name data are generalizable. In addition, we would like to find out if models trained using the smaller SSA DATA data are extendable as well.

For these experiments, we split SSA DATA into an 80/20 ratio, corresponding to train/test sets. To test the generalizability properly, we took out all names in SSA DATA test data that existed in YAHOO DATA train set, leading to 2,077 remaining names in SSA DATA test set.

Table 6 gives the results of our five models. We trained using the train sets for YAHOO DATA and SSA DATA, and tested on the test sets for YAHOO DATA and SSA DATA, respectively.

Training on Yahoo Data and testing on SSA Data: from Table 6 (top), we see that LSTM, char-BERT and NBLR generally work the best, and their performance is consistent with their results reported in Table 4. NBLR (AUC = 0.972) is slightly worse than LSTM and char-BERT (AUC = 0.980). However the differences are very small, indicating that NBLR works effectively for predicting user's genders. Importantly, all our models perform very well on SSA test data, indicating that *our models trained on* Yahoo Data *generalize well to* SSA Data.

Training on SSA Data and testing on SSA Data: from Table 6 (middle), we see that when trained and tested using the SSA Data, NBLR and LSTM perform the best, but others also work reasonably well, indicating that SSA Data, as an English only dataset, is homogeneous and relatively easy to predict.

Training on SSA Data and testing on Yahoo Data: In Table 6 (bottom), we build models using the smaller SSA Data train set and evaluate on large-scale Yahoo Data test set. All models perform worse compared to their corresponding performance reported in Table 4 (i.e. trained on Yahoo Data train data and evaluated on Yahoo Data test data). This indicates that *building models using the smaller* SSA Data *train data does not generalize well to the larger* Yahoo Data *at all*. For example, NBLR only achieves AUC of 0.859 on the Yahoo Data test set, vs NBLR trained using Yahoo Data, which achieved AUC of 0.940 in Table 4. This is not a limitation of our models, but due to the fact that SSA Data is an English based dataset, so models trained using it could not be expected to understand the non-English/multilingual names in Yahoo Data. We note that another reason that Yahoo Data seems to be harder to predict than SSA Data is due to the presence of noise. The absence of noise in the latter also makes it easier to classify.

Overall, we conclude that models trained using the larger Yahoo Data extend very well to names that are not observed in the train data.

5 Using both first and last names for gender classification

While the gender is typically conveyed in the first name, there are cases when the same first name can be used for one gender in one culture, but another in a different culture. For example, the Italian male name "Andrea" (derived from the Greek "Andreas") is considered a female name in many languages, such as English, German, Hungarian, Czech, and Spanish (Wikipedia: Unisex name 2020). Since the last name often conveys a person's ethnicity, we speculate that it may be beneficial to include it in the gender classification for such unisex names. In this section we attempts to answer this conjecture.

5.1 A Dual-LSTM model that utilizes both the first and the last names

While the linear classifier NBLR was found to be simple yet effective for predicting gender using first names, it is not suited for utilizing full names. One way to include the last name in a linear model would be to construct features that conjunct character-based n-gram features for both first and last names. However, doing so would increase

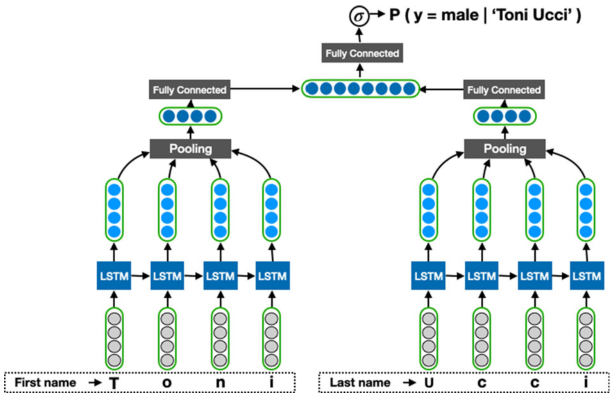


Fig. 3 The dual-LSTM (DLSTM) model that utilizes both first and last names

the feature cardinality quadratically. Therefore it is necessary to explore different approaches.

We propose to encode each of the first and last names into a dense vector via character LSTM, then predict the gender using a concatenation of the two vectors. We call this the dual-LSTM model, or DLSTM for short. Figure 3 illustrates the model architecture.

For comparison, we propose another approach using name embeddings (Han et al. 2017; Ye et al. 2017) via the EMB algorithm. As explained in Sect. 3.7, both the first and last name are embedded into a 100-D space by applying word2vec to the contact lists of millions of users. We then encode a full name into a 200-D vector by concatenating their respective embeddings, and apply XGBoost to predict the gender.

5.2 Results of models using both first and last names

To conduct this experiment, we used Yahoo Full Name Data, which contains the first and last names of 13M users. We split the data into 70% train, 10% validation and 20% test sets, making sure that the same full name only appears in one of these 3 sets. We also experimented with a 5-fold cross validation, and found that the variance for AUC and ACC were less than 0.001. Therefore, we only report the mean results.

For DLSTM, each character is embedded as a 256 dimensional vector, before passing through LSTM units. We experimented with adding an attention layer in place of the pooling layer, but did not find it helpful. Instead, as in LSTM, we find that using the embeddings of the last characters work well. We fed the last embeddings to a dense layer of 300 neurons, then concatenated the output into a 600-D vector. We found that it is necessary to have fully connected layers after the concatenation, in order to allow the two embeddings to interact properly. We used fully connected layers with 512, 256, 64 neurons, before feeding to a sigmoid function that gives the final predicted probability. We limited the max characters to 30 per arm of the DLSTM, and pad empty space in front for short names. To make a consistent comparison, the LSTM

Table 7 Performance of DLSTM and EMB (with XGBoost) using both first and last names, vs LSTM, NBLR and EMB with only first names. Testing is on the test set of the YAHOO FULL NAME DATA, or on the ANDREA DATA and the TONI DATA

		YAHOO FULL NAME DATA		ANDREA DATA		TONI DATA	
Model	Features	AUC	ACC	AUC	ACC	AUC	ACC
LSTM	first	0.949	0.889	0.509	0.648	0.506	0.556
LSTM	full string	0.953	0.895	0.841	**0.827**	0.826	0.792
DLSTM	first+last	**0.957**	**0.901**	0.823	0.796	**0.886**	**0.850**
EMB	first	0.941	0.871	0.5	0.506	0.5	0.648
EMB	first+last	0.946	0.879	**0.857**	0.783	0.854	0.811
NBLR	first	0.951	0.891	0.5	0.556	0.5	0.352

model in this section follows a similar setup (we find that it performed similarly to the LSTM in the previous sections).

Table 7 shows the results of the LSTM and DLSTM classifiers. For comparison, we also give the corresponding results when using the name embedding algorithm EMB. It is seen that DLSTM that uses both the first and last names improves the AUC/ACC by 0.8%/1.1% respectively, when compared with LSTM that uses first name only, indicating that information from the last name is helpful. LSTM model also works better than the EMB model.

As an additional test, we also tried a simpler way of utilizing the full names: we use the first and last names as a single string separated by a space (shown as "LSTM/full string" in Table 7), and applying the LSTM algorithm (Figure 2 left). This simpler way of utilizing first and last names does not perform as well as DLSTM, even though it is a little better than LSTM using first name alone (shown as "LSTM/first" in Table 7).

For a fair comparison, we also applied NBLR to first names. We give its performance in Table 7. It does perform similarly to LSTM with first names, but as discussed earlier it is not suitable for taking advantage of both first and last names. It is also notable that both LSTM and NBLR give better performance on this full name dataset than on YAHOO DATA (Table 4). This is because YAHOO FULL NAME DATA is sampled based on full names, as such the first names in this data set tend to be more popular. Consider the population of users in YAHOO DATA, we found that first names in YAHOO DATA have an average of only 20 users, versus 704 for YAHOO FULL NAME DATA. We knew from Table 4 that it is easier to classify the gender of popular names. We did try applying NBLR to full names by concatenating first and last names with a special character separator, the result is even worse than using first names alone.

We wanted to see if the LSTM model trained using both the first and the last names was especially effective for unisex first names. We knew that unisex first names are not as common as gendered first names. For example, about 11% of the Social Security baby names (Social Security Administration 2018) are documented as both male and female. Given the relative small percentage of unisex names, the effect of a model on these names may be diluted when looking at the overall performance in Table 7.

We divide users in YAHOO FULL NAME DATA into 40 buckets based observed $P(M|\text{first})$ (first name appears less than 10 times are filtered out). We denote these

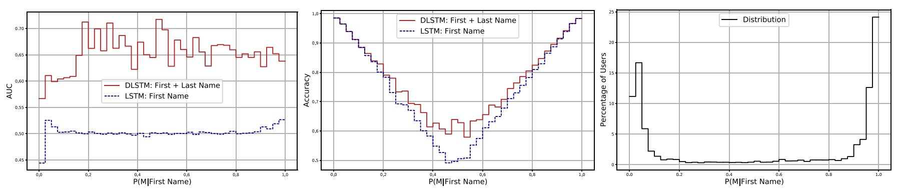


Fig. 4 The AUC (left) and ACC (middle) of DLSTM vs LSTM, as a function of how unisex the first names are. People are divided into 40 buckets on the x-axis based on $P(\text{M}|\text{first})$. Names in the middle of the x-axis, away from 0 and 1, are unisex names. The distribution of people are show on the right

buckets $b[i]$, $(i = 1, 2, \ldots, 40)$. Specifically, $b[i] = [0.025 * (i - 1), 0.025 * i)$. The first few buckets are for people with first names that are used almost exclusively by women. E.g., $b[1] = [0, 0.025)$ contains people with a first name that are used by men only $< 2.5\%$ of the time. The last few buckets are people with first names that are used almost exclusively by men. E.g., $b[40] = [0.975, 1.0]$ contains people with a first name that are used by men $\geq 97.5\%$ of the time. The middle buckets contain people with unisex names. For example, $b[20]$ and $b[21]$ contains people with first names used by men between 47.5% to 52.5% of the time.

Figure 4 (right) shows the distribution of people in the 40 buckets. In all plots in Fig. 4, the x-axis, from left to right, corresponds to buckets 1 to 40. Clearly few people belongs to the unisex buckets. Figure 4 (left/middle) shows the AUC/ACC comparison of LSTM that uses first name only, vs DLSTM that uses both first and last names, on each of the 40 buckets. As can be seen from Fig. 4 (middle), for the first and last few buckets, which correspond to people with first names that are exclusively used by women or by men, the ACC difference between these two algorithms are relatively small. The gap widens to about 10% as we moved to the middle area that contains unisex names. When looking at AUC, LSTM has AUC values around 50%, indicating that whatever the percentage of women/men a bucket has, using first name alone is not enough to tell them apart. DLSTM, which uses both first and last names, consistently outperform with 10-15% higher AUC. This confirms that using full names is beneficial, especially when classifying the gender of unisex names.

To given some concrete examples, we collected two unisex name datasets. We found 133 famous people (74 women, 59 men) with the first name "Andrea" (Wikipedia: Andrea 2020), and call this the Andrea Data. In addition we found 54 famous people (19 women, 35 men) with the first name "Toni" (Wikipedia: Toni 2020), and call this the Toni Data. About 16% of "Andrea" and 33% of "Toni" in Yahoo Full Name Data are men. To avoid overfitting, we removed any person in Yahoo Full Name Data whose full name matches those 187 names in Andrea Data and Toni Data. Table 7 shows that on these two datasets, the DLSTM model is much more effective than the LSTM model with only first names. On the Andrea Data, AUC improved from 0.509 to 0.823, while on the Toni Data, AUC improved from 0.506 to 0.886.

Here we give some examples where the full name based model works well. For men, "Andrea Bianchi" (former Italian film director and writer), "Andrea Boattini" (Italian astronomer), "Andrea Bargnani" (Italian former professional basketball player); for women, "Andrea Sestini Hlaváčková" (Czech professional tennis player), "Andrea Růžičková" (Slovak actress and model), "Andrea Murez" (Israeli-American Olympic swimmer).

There are cases where the model failed, including, for men misclassified as women, "Andrea Tacquet" (Belgium Mathematician) and "Andrea Costa" (Italian Master Mason and socialist activist); and for women misclassified as men, "Andrea Gyarmati" (Hungarian swimmer) and "Andrea Fioroni" (Argentine field hockey player). Given that "Fioroni" is a family name originated from Italy, this last failure can actually be considered an anomaly in the data.

6 Related work

Given the importance of gender information for ad targeting, unsurprisingly there are many existing works on gender classification.

6.1 Content based gender classification

Grbovic et al. (2015) studied gender based ad targeting in Tumblr. Their model created gender labels using SSA Data as the ground truth. It then uses blog content (titles, texts) and user actions (follows, likes and reblogs) as features, much like the content model described in Sect. 2.3.

Many works infer the user gender by extracting features based on user activity and user-generated content on social media (Kokkos and Tzouramanis 2014; Ciot et al. 2013; Rao and Yarowsky 2010; Pennacchiotti and Popescu 2011; Otterbacher 2010; Burger et al. 2011; Liu et al. 2012; Al Zamal et al. 2012; Culotta et al. 2015, 2016; Merler et al. 2015; Sakaki et al. 2014; Beretta et al. 2015; Ludu 2014). Otterbacher (2010) inferred gender of 21k movie reviewers by their writing style, content and metadata. Ciot et al. (2013) inferred gender of 8.6k Twitter users by the top-k discriminative language features between male and female, such as the top-k hashtags, words, mentions, *etc.* Liu et al. (2012) used Twitter content to infer gender of 33k users. Merler et al. (2015) used pictures coming from the user's feeds to infer gender of 10k Twitter users. Sakaki et al. (2014) combined both user-generated text and images in their feeds to infer gender of 3.9k Twitter users.

Unlike these works, we only infer users' gender using their names. This allows us to immediately infer the gender of newly registered users without requiring any user activity or content, and avoids the issue of sparse signals in content-based models.

6.2 Name based demographic classification

Several works predicted gender based on users' names (Knowles et al. 2016; Mueller and Stumme 2016). Liu and Ruths (2013) proposed two methods of incorporating names and other Twitter features for predicting gender of Twitter users. They found that using the gender association score of a name as a switch to determine whether to bypass the content-based classifier works better than using both kinds of features.

Gender association score was computed using name distributions from the 1990 US census. However, no name classifier was involved.

Ambekar et al. (2009) used full names to infer ethnicity, using hidden Markov models. Han et al. (2017) found evidence of ethnic/gender homophily in email correspondence patterns. Through large-scale analysis of contact lists, they found that there was a greater than expected concentration of names of the same gender and ethnicity within the same contact lists. Consequently, they generated name embeddings using word2vec, and used the embeddings for gender, ethnicity and nationality classification with good results (Han et al. 2017; Ye et al. 2017).

Brown (2017) proposed a naive Bayes classifier to classify the gender of a name. The features used are simpler than ours and came from an `nltk` book: first/last letter, count of letters, has letters, suffixes (last 2, 3, 4 letters of name). They also compared with an RNN character-based model and found that the Naive Bayes model performs much better. Compared with their method, on the same dataset (Brown 2017), we found our NBLR classifier to be more accurate (0.878 vs. 0.85 ACC.)

Overall, our work is different from previous works in five areas: (1) We studied the gender prediction task on two much larger scale datasets: the Yahoo Data with 21M unique first names and the SSA Data with 98.4k unique names; (2) we built many character-based models, from the NBLR linear model to the advanced char-BERT model, and conduct a comprehensive analysis on these models; (3) we verified the effectiveness of our proposal on multilingual names and its generalizability for unseen datasets; (4) we compared with content and embedding based baselines and showed that our approach is more accurate; (5) we demonstrated that the performance of character-based machine learning models could be further enhanced when utilizing both the first and last names, as well as the content the user interacted with (see Appendix).

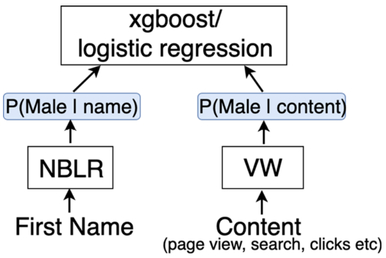
7 Conclusion

In this paper, we considered the problem of inferring gender from users' first names. We proposed character-based machine learning models, and demonstrated that our models are able to infer the genders of users with much higher accuracy than models based on content information, or based on name embeddings. In addition, we find surprisingly that a linear model, with careful feature engineering, performed just as well as complex models such as LSTM or char-BERT. We verify the generalizability of our proposal for unseen datasets.

First names may have different gender connotations in different cultures. We demonstrated in Sect. 5 using a dual-LSTM model that utilizing both first and last names can be effective in such a situation.

Finally, while a content based model offers inferior performance to the name based models, we believe that it could be complementary to the latter, especially for people with unisex names. We conducted a study in Sect. A.1 showing that an ensemble model that combines both first name and content information gives a further performance boost.

Fig. 5 Fusion model that combines name and content information to predict gender



One limitation of using names as the main signal for classifying genders is that users may not disclose their true names. Another limitation is that gender may be difficult to infer if the first name is unisex. We showed separately that it is helpful to use the first and last names, or the first name and content information. For future work, we intend to investigate whether combining these all together (first and last names, and content information) can give us even better performance.

Acknowledgements The authors like to thank the reviewers for the very thoughtful reviews of earlier versions of this paper.

A Appendix

A.1 Fusion model that combines content information and names for gender prediction

We have seen in Sect. 4 (Table 4) that the content based model is inferior to the name based models. However, it is reasonable to expect that because content consumption by users captures behavior signals, it may offer useful information that can complement the name based models. This is especially true if the user has a unisex first name, for which it is not possible to predict with certainty the gender of the user by first name alone.

We conducted a study to investigate a simple ensemble model. We took a sample of 1M users from YAHOO FULL NAME DATA. For each user, we predicted the male probability using the first name and the NBLR model. For the same user, we predicted the probability based on the content the user consumed, using the CONTENT model.

We experimented with the following ways of fusing the results of the content and name based models:

Logistic regression: we use $P(\text{M}|\text{first})$ and $P(\text{M}|\text{content})$ as features and declared gender as labels to fit a logistic regression model.

Logistic regression with logits: we first convert $P(\text{M}|\text{first})$ and $P(\text{M}|\text{content})$ to logits, then fit a logistic regression model. Specifically, we convert the probability to logits via the function $p \rightarrow \ln(p/(1-p))$.

Xgboost: we use $P(\text{M}|\text{first})$ and $P(\text{M}|\text{content})$ as features and declared gender as labels to fit an xgboost model.

We did a 5-fold cross-validation to derive the average results in Table 8. We do not report the standard deviation since it is quite small (under 0.0006). From the table,

Table 8 Performance comparison of 3 fusion models, vs the baseline name and content models

Model	Feature	AUC	ACC
logistic regression	first name+content	<u>0.9764</u>	<u>0.9333</u>
logistic regression with logit	first name+content	0.9762	0.9319
xgboost	first name+content	**0.9784**	**0.9346**
name (NBLR)	first name	0.9709	0.9230
content	content	0.7937	0.7270

we see that the fusion model using xgboost can improve AUC by 0.008, accuracy by 0.012, confirming our belief that combining the content and names improves gender prediction. Logistic regression performed slightly worse than xgboost. Logistic regression with logits does not perform as well as logistic regression, although still better than the name based model alone.

A.2 Dictionary look-up vs model prediction

For users with popular names, it is unnecessary to use a name-based machine learning model to predict their genders. Instead, a simple dictionary lookup is sufficient – it avoids any error that might be introduced in the modeling process. On the other hand, for unpopular names, a model-based method is more suitable: when a name appears only a few times, statistically we have a low confidence in deciding the gender based on the few occurrences. Furthermore, for names that have never appeared before in the data, a machine learning model is the only choice.

Therefore, when deploying in real world applications, we suggest taking a hierarchical approach for the gender classification: if a name appears $\geq k$ times in the population of users with declared gender, we decide the gender based on a dictionary lookup. For other names, we apply the machine learning based models. We will reason on a suitable choice of k next.

In general, we want to pick a name frequency k, above which we can trust the labels in the lookup dictionary with a high confidence. Suppose that a name is 30%/70% male/female in a very large population (e.g., the world). If there are k of our users with that name, drawn from that population randomly, then based on simple calculation using binomial distribution, we need at least $k \geq 16$ to be sure that the name will be labelled as female with 95% or higher probability [3]. For a name that is 40%/60% male/female, we need $k \geq 66$. Based on the above considerations, we decide to choose $k = 40$. With that name frequency, we know from Table 2 that a dictionary lookup can cover the gender of 86% of users. The gender for remaining users can be inferred by the machine learning models. *We emphasize that dictionary lookup is proposed only for real-world applications. For the purpose of understanding our models for names*

[3] Given a predominantly female name with a male probability of $p < 0.5$, and given k randomly selected people with this name, the probability that more than half of these people are male is: $f(p, k) = \sum_{i=\lceil k/2 \rceil)}^{k} C_n^i p^i (1-p)^{k-i}$. For $p = 0.3$, we found that $f(p, k) < 0.05$ when $k \geq 16$. For $p = 0.4$, we found that $f(p, k) < 0.05$ when $k \geq 66$.

of different frequencies, all results reported in this paper are based on the machine learning models. We have seen in Table 4 that our proposed models performed well for both popular and unpopular names, when compared with the baselines.

A.3 Understanding the models

In the previous subsection, we have seen that NBLR, LSTM and char-BERT all perform very similarly. Normally, a linear model such as NBLR is easier to understand than nonlinear models. By looking at the weights for the features, one can understand why a sample is predicted as positive or negative.

However, in the case of predicting the gender using the name string, the features are character n-grams extracted from the same name, hence they are not independent. This feature correlation can give rise to negative weights for some features that are associated with male names, which is compensated by larger positive weights for other correlated features (as a reminder, we use label 1 for male and 0 for female). Similarly positive weights may be observed for features that are associated with female names. This makes model interpretation from feature weights difficult. For example, when looking at the female name "_anna_" (we add prefix and suffix "_" to a name, see Sect. 3.2), we found that the 5-gram "anna_" has a positive weight of 0.36, which may be a surprise as one would expect that a name ends with "anna" should be a female name. However, a substring of "anna_" is "nna_", and it has a negative weight of -1.19. When combining the weights of all 17 character grams, together with the intercept, we end up with a weight of -9.01 and a male probability of close to 0. Thus, looking at feature weights can be confusing and even misleading.

Instead, it is more instructive to apply one of our models to real datasets, and see how it behaves. We first apply NBLR to classify Chinese names. Note that the Chinese language has up to 100K characters. Of these, about 7000 are commonly used. A person's given name could consist of any one or two characters. Therefore, the number of unique given names could be in the millions ($7000 \times 7000 = 49$M). It is difficult to collect enough instances of each of these possible variations and their associated gender. This is where a character-based model is especially suited. As a sanity check, we looked at a list of 11 names from Lu (2018) that are considered especially beautiful, because of their poetic connotation or phonetic harmony. Table 9 shows the male probabilities for each of these names, together with a "note" column for their likely genders and explanation, taken from Lu (2018). We observe that our classifier does assign higher scores for names that are definitely male than those that are female.

It is also helpful to understand the NBLR classifier by looking at English words that scored high as male or female. We first take the top (3000 most common words in 3000 most common words in english 2020), filter out 443 words that are found in SSA Data (such as "Rose"), then list the top 11 highest scored male/female words in Table 10 (a). Interestingly, a lot of them make sense, e.g., "miss" or "women" are female words due to their meanings. However, we notice that the majority of these 3,000 words are found in Yahoo Data as names. Next, we take 236,736 words in `nltk.corpus.words` and filter out any names in either SSA Data or Yahoo

Table 9 "Eleven Beautiful Chinese Names" from Lu (2018). $P(M)$ is the probability of male, predicted by NBLR. The "label" column gives the likely gender, and the "note" column gives a description, both based on Lu (2018). Three of the eleven has frequency $\geq$ 40 and their genders (in brackets) could have been looked up

name	$P(M)$	label	note
芷若(F)	0.00	F	Name for girls of two herbal plants
语嫣	0.00	F	Popular female character by Jin Yong
映月	0.04	F	Name for girls. "reflection of the moon"
念真(F)	0.12	M/F	A belief in truthfulness
徽因	0.26	F	Famous female Chinese poet/architect
望舒	0.29	M/F	God who drives the moon carriage
如是	0.30	F	Famous Chinese courtesan and poet
莫愁	0.64	M/F	Free of sadness
风眠	0.77	M/F	Falling asleep in the woods/breeze
青山(M)	0.85	M	Blue-coloured mountains
飞鸿	0.87	M	A swan goose soaring high in the sky

DATA, and end up with 169,902 words. The highest scoring masculine/feminine words are given in Table 10 (b). Many of the words in column "feminine" of Table 10 (b) do look very feminine. For example, many words end in "ia" (e.g., "anadenia"). Words on the right-hand-side of the table are classified as masculine, likely due to the ending of the words, such as "*boy". Incidentally "knappan" is a word in Malayalam (a Dravidian language primarily spoken in the south Indian state of Kerala) that means "a good for nothing guy".

Table 10 (c) gives the most feminine and masculine company names in S&P 500 companies (SP 500 Companies 2020). The results are also interesting. Corporations with feminine names include a number of ladies fashion companies (Tiffany, Ulta, Macy's), while many with masculine names are in engineering or energy sectors (e.g., Duke Energy, General Motor, General Electric).

Overall, the performance of NBLR on these three datasets illustrates that the model can extend well to rare or even unseen strings, and to non-English languages.

Table 10 Words with a high female/male score among: (a) 3,000 most frequent English words that are not in SSA DATA; (b) 236,736 words in `nltk.corpus.words` that are neither in SSA DATA nor in YAHOO DATA. (c) S&P 500 companies

(a) Top 3,000 words		(b) `nltk.corpus.words`	
feminine	masculine	feminine	masculine
miss	gentleman	Myrianida	unclement
woman	father	nonylene	farcial
Mrs	engineer	decylene	maccoboy
makeup	commander	shamianah	comradery
criteria	military	whidah	knappan
guideline	hard	Manettia	subramous
nurse	big	anadenia	herdboy
pregnancy	auto	mollienisia	beshod

Table 10 continued

(a) Top 3,000 words		(b) nltk.corpus.words	
feminine	masculine	feminine	masculine
mom	dad	Anacanthini	Geophilus
daughter	kill	phenylene	clocksmith
wife	electronic	vinylene	Bavius

(c) S&P 500 companies

feminine	masculine
Tiffany & Co.	Emerson Electric
Ulta Beauty	Cisco Systems
Kimberly-Clark	Philip Morris International
DaVita Inc.	Raymond James Finance Inc.
Apple Inc.	Henry Schein
Nike	Mohawk Industries
Danaher Corp.	Duke Energy
CIGNA Corp.	General Motor
Lilly (Eli) & Co.	Stanley Black & Decker
Macy's Inc.	Edison Int'l
Dentsply Sirona	General Electric

References

3000 most common words in english. https://www.ef.edu/english-resources/english-vocabulary/top-3000-words/ (2020). [Online; accessed March 22, 2020]

SP 500 Companies (2020). https://datahub.io/core/s-and-p-500-companies. [Online; accessed March 22, 2020]

Social Security Administration: National data on the relative frequency of given names in the population of U.S. births where the individual has a social security number (tabulated based on social security records as of march 3, 2019). http://www.ssa.gov/oact/babynames/names.zip

Al Zamal F, Liu W, Ruths D (2012) Homophily and latent attribute inference: Inferring latent attributes of twitter users from neighbors. In: Sixth International AAAI Conference on Weblogs and Social Media

Ambekar A, Ward C, Mohammed J, Male S, Skiena S (2009) Name-ethnicity classification from open sources. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge Discovery and Data Mining, pp. 49–58. ACM

Beretta V, Maccagnola D, Cribbin T, Messina E (2015) An interactive method for inferring demographic attributes in twitter. In: Proceedings of the 26th ACM Conference on Hypertext & Social Media, pp. 113–122. ACM

Brown E (2017) Gender inference from character sequences in multinational first names. https://towardsdatascience.com/name2gender-introduction-626d89378fb0#408a

Burger JD, Henderson J, Kim G, Zarrella G (2011) Discriminating gender on twitter. In: Proceedings of the conference on empirical methods in natural language processing, pp. 1301–1309. Association for Computational Linguistics

Chen P, Sun Z, Bing L, Yang W (2017) Recurrent attention network on memory for aspect sentiment analysis. In: Proceedings of the 2017 conference on empirical methods in natural language processing, pp. 452–461

Ciot M, Sonderegger M, Ruths D (2013) Gender inference of twitter users in non-english contexts. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1136–1145

Google Cloud Content Categories (2019). https://cloud.google.com/natural-language/docs/categories

Culotta A, Kumar NR, Cutler J (2015) Predicting the demographics of twitter users from website traffic data. In: AAAI, pp. 72–78

Culotta A, Ravi NK, Cutler J (2016) Predicting twitter user demographics using distant supervision from website traffic data. J Artif Intell Res 55:389–408

Devlin J, Chang MW, Lee K, Toutanova K (2018) Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805

Grbovic M, Radosavljevic V, Djuric N, Bhamidipati N, Nagarajan A (2015) Gender and interest targeting for sponsored post advertising at tumblr. In: proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15, pp. 1819–1828. ACM, New York, NY, USA. https://doi.org/10.1145/2783258.2788616

Han S, Hu Y, Skiena S, Coskun B, Liu M, Qin H, Perez J (2017) Generating look-alike names for security challenges. In: proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec '17, pp. 57–67. ACM, New York, NY, USA. https://doi.org/10.1145/3128572.3140441

Hochreiter S, Schmidhuber J (1997) Long short-term memory. In: neural Computation, pp. 1735–1780

Karako C, Manggala P (2018) Using image fairness representations in diversity-based re-ranking for recommendations. In: adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization, pp. 23–28. ACM

Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980

Knowles R, Carroll J, Dredze M (2016) Demographer: Extremely simple name demographics. In: proceedings of the First Workshop on NLP and Computational Social Science, pp. 108–113

Kokkos A, Tzouramanis T (2014) A robust gender inference model for online social networks and its application to linkedin and twitter. First Monday 19(9)

Liu W, Al Zamal F, Ruths D (2012) Using social media to infer gender composition of commuter populations. In: sixth international AAAI Conference on Weblogs and Social Media

Liu W, Ruths D (2013) What's in a name? using first names as features for gender inference in twitter. In: analyzing microtext AAAI 2013 Spring Symposium, pp. 10–16. AAAI, Palo Alto, CA, USA

Lu F (2018) The 11 Most Beautiful Chinese Names and What They Mean. https://bit.ly/2yGSNO7

Ludu PS (2014) Inferring gender of a twitter user using celebrities it follows. arXiv preprint arXiv:1405.6667

Merler M, Cao L, Smith JR (2015) You are what you tweet...pic! gender prediction based on semantic analysis of social media images. In: 2015 IEEE International Conference on Multimedia and Expo (ICME), pp. 1–6. IEEE

Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. In: proceedings of Workshop at ICLR

Mueller J, Stumme G (2016) Gender inference using statistical name characteristics in twitter. In: proceedings of the The 3rd Multidisciplinary International Social Networks Conference on SocialInformatics, Data Science 2016, p. 47. ACM

Otterbacher J (2010) Inferring gender of movie reviewers: exploiting writing style, content and metadata. In: proceedings of the 19th ACM international conference on Information and knowledge management, pp. 369–378. ACM

Pennacchiotti M, Popescu AM (2011) A machine learning approach to twitter user classification. In: Fifth International AAAI Conference on Weblogs and Social Media

Rao D, Yarowsky D (2010) Detecting latent user properties in social media. In: Proc. of the NIPS MLSN Workshop, pp. 1–7. Citeseer

Sakaki S, Miura Y, Ma X, Hattori K, Ohkuma T (2014) Twitter user gender inference using combined analysis of text and image processing. In: proceedings of the Third Workshop on Vision and Language, pp. 54–61

Wang S, Manning CD (2012) Baselines and bigrams: Simple, good sentiment and topic classification. In: proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2, ACL '12, pp. 90–94. Association for Computational Linguistics, Stroudsburg, PA, USA

Wang Y, Huang M, Zhao L, et al. (2016) Attention-based lstm for aspect-level sentiment classification. In: proceedings of the 2016 conference on empirical methods in natural language processing, pp. 606–615

Wikipedia: Andrea. https://en.wikipedia.org/wiki/Andrea [Online; accessed March 22, 2020]

Wikipedia: Toni. https://en.wikipedia.org/wiki/Toni [Online; accessed March 22, 2020]

Wikipedia: Unisex name. https://en.wikipedia.org/wiki/Unisex_name [Online; accessed March 22, 2020]

Wu Y, Schuster M, Chen Z, Le QV, Norouzi M, Macherey W, Krikun M, Cao Y, Gao Q, Macherey K, Klingner J, Shah A, Johnson M, Liu X, Łukasz Kaiser, Gouws S, Kato Y, Kudo T, Kazawa H, Stevens

K, Kurian G, Patil N, Wang W, Young C, Smith J, Riesa J, Rudnick A, Vinyals O, Corrado G, Hughes M, Dean J (2016) Google's neural machine translation system: Bridging the gap between human and machine translation. CoRR arXiv:1609.08144

Yao S, Huang B (2017) Beyond parity: Fairness objectives for collaborative filtering. In: advances in neural information processing systems, pp. 2921–2930

Ye J, Han S, Hu Y, Coskun B, Liu M, Qin H, Skiena S (2017) Nationality classification using name embeddings. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM '17, pp. 1897–1906. ACM, New York, NY, USA. https://doi.org/10.1145/3132847.3133008

Zhang X, Zhao J, LeCun Y (2015) Character-level convolutional networks for text classification. In: proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, NIPS'15, pp. 649–657. MIT Press, Cambridge, MA, USA

Zhou X, Wan X, Xiao J (2016) Attention-based lstm network for cross-lingual sentiment classification. In: proceedings of the 2016 conference on empirical methods in natural language processing, pp. 247–256