Reproducing Comparison and benchmark of name-to-gender inference services

David Arroyo Menéndez

January 22, 2019

Abstract

- Authors: Lucía Santamaría (Amazon) and Helena Mihaljević (University of Applied Sciences)
- Compare and benchmark five name-to-gender inference services by applying them to the classification of a test data set consisting of 7,076 manually labeled names.
- The compiled names are analyzed and characterized according to their geographical and cultural origin.
- We define a series of performance metrics to quantify various types of classification errors, and define a parameter tuning procedure to search for optimal values of the services' free parameters.

Comparison and benchmark of name-to-gender inference services (I)

Table 1 **Comparison table showing relevant features for the gender inference services under study.** Note that although Gender API does provide a specific API end point for handling surnames, our results employ the version that does not make use of them.

	Gender API	gender-guesser	genderize.io	NameAPI	NamSor
Database size (January 2018)	1,877,787	45,376	216,286	510,000	1,300,000
Regular data updates	yes	no	yes	yes	yes
Handles unstructured full name strings	yes	no	no	yes	no
Handles surnames	yes	no	no	yes	yes
Handles non-Latin alphabets	partially	no	partially	yes	yes
Implicit geo-localization	no	no	no	yes	yes
Assignment type	probabilistic	binary	probabilistic	probabilistic	probabilistic
Free parameters	accuracy, samples	–	probability, count	confidence	scale
Open source	no	yes	no	no	no
API	yes	no	yes	yes	yes
Monthly free requests	500	unlimited	30,000	10,000	1,000
Monthly subscription cost (100,000 requests/month)	79 €	Free	7 €	150 €	80 €
Provider	Gender-API.com	Israel Saeta Pérez	Casper Strømgren	Optimaize GmbH	NamSor SAS

Comparison and benchmark of name-to-gender inference services (I)

Database size	damegender
Regular data updates	yes, developing
Handles unstructured full name strings	yes
Handles surnames	yes
Handles non-Latin alphabets	no
Implicit geo-localization	no
Assingment type	probabilistic
Free parameters	ml
Free license	yes
API	future
free requests limited	unlimited

Comparison and benchmark of name-to-gender inference services (II): Assembling data

- zbMath: names from articles, labeled by humans using university websites, Wikipedia articles, etc.
- genderizeR: from full names searching in Internet biographies, personal data, etc.
- PubMed: six highest JCR
- WoS: from 2008 to 2012 included in Science Citation Index Expanded, the Social Sciences Citation Index, and the Arts and Humanities Citation Index

Origin of the names (I)

Table 2 **Examples of the geographical origins of names as inferred by NamSor's origin API.**

Full_name	Gender	Source	Country	Top_region	Sub_region	Score
maria bortolini	f	wos	Italy	Europe	Southern Europe	2.925873
liew woei kang	m	pubmed	China	Asia	Eastern Asia	2.638786
sirin yasar	f	wos	Turkey	Asia	Western Asia	3.357177

Origin of the names (II)

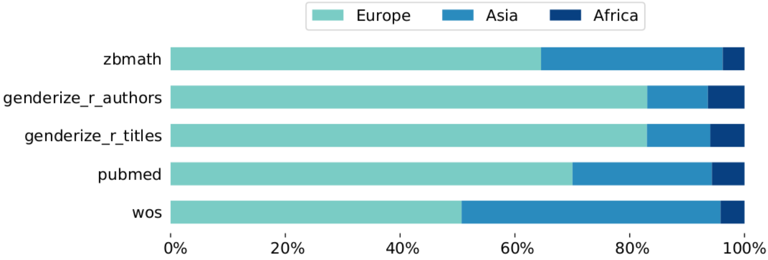


Figure 1 **Geographical region of origin of the personal names from our test data set as inferred by NamSor's origin API.** The colored bars show percentages split by data sources. The genderize_r data sets are the most Eurocentric, whereas the wos collection is more balanced towards Asian names. African names amount to at most 6% per data source, which reflects the shortage of scholarly authors from that region.

Full-size DOI: 10.7717/peerjcs.156/fig-1

Retrieval of gender assignments

```
$ python3 api2gender.py David --api=genderguesser
male
```

Performance metrics (I): accuracy method

```
def accuracy_score_dame(self, v1, v2):
    if (len(v1) == len(v2)):
        success = 0
        fails = 0
        for i in range(0, len(v1)):
            if (v1[i] == v2[i]):
                success = success + 1
            else:
                fails = fails + 1
        if (fails == 0):
            accuracy = 1
        else:
            accuracy = success / len(v1)
    else:
        accuracy = 0
        print("Both vectors must have the same length")
    return accuracy
```

Performance metrics (II): executing accuracy.py

```
$ python3 accuracy.py --csv=files/min.csv
files/min.csv
#################### Namsor!!
Gender list: [1, 1, 1, 1, 2, 1, 0, 0]
Guess list:  [1, 1, 1, 1, 1, 1, 0, 0]
0.875
Namsor accuracy: 0.875
#################### Genderize!!
Gender list: [1, 1, 1, 1, 2, 1, 0, 0]
Guess list:  [1, 1, 1, 1, 2, 1, 0, 0]
Genderize accuracy: 1
#################### GenderGuesser!!
Gender list: [1, 1, 1, 1, 2, 1, 0, 0]
Guess list:  [1, 1, 1, 1, 2, 1, 0, 0]
GenderGuesser accuracy: 0.875
#################### Sexmachine!!
Gender list: [1, 1, 1, 1, 2, 1, 0, 0]
```

Confusion (I): table

		predicted class		
		male	female	unknown
true class	male	m_m	m_f	m_u
	female	f_m	f_f	f_u

Confusion (II): male male

```
def malemale(self, truevector, guessvector):
    i = 0
    count =0
    maxi = len(truevector)
    while (i < maxi):
        if ((truevector[i]==1) and (guessvector[i]==1)):
            count = count + 1
        i = i +1
    return count
```

Confusion (III): Matrices

Table 3 **Confusion matrices for all services using their default responses without parameter tuning.**

	m_{pred}	f_{pred}	u_{pred}
(a) Gender API			
m	3,573	110	128
f	172	1,750	46
(b) gender-guesser			
m	2,964	66	781
f	56	1,530	382
(c) genderize.io			
m	3,210	189	412
f	73	1,744	151
(d) NameAPI			
m	3,126	93	592
f	75	1,616	277
(e) NamSor			
m	3,354	132	325
f	94	1,684	190

Confusion (IV): damegender

```
 $ python3 confusion.py
A confusion matrix C is such that Ci,j is equal to the number o
If the classifier is nice, the diagonal is high because there a
Namsor confusion matrix:
 [[2 0 0]
 [0 5 0]
 [0 1 0]]
Genderize confusion matrix:
 [[2 0 0]
 [0 5 0]
 [0 0 1]]
Gender Guesser confusion matrix:
 [[2 0 0]
 [0 5 0]
 [0 1 0]]
Sexmachine confusion matrix:
 [[2 0 0]
```

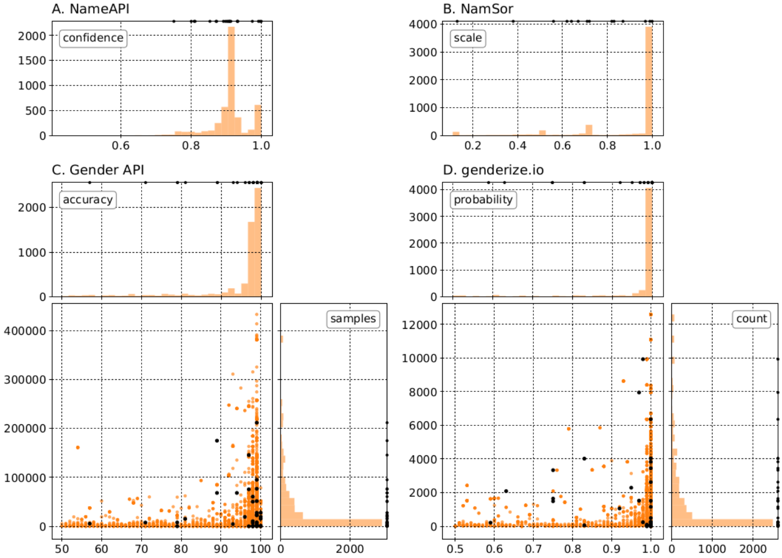
Errors

$$\begin{aligned}
\text{errorCoded} &= \frac{f_m + m_f + m_u + f_u}{m_m + f_m + m_f + f_f + m_u + f_u}, \\
\text{errorCodedWithoutNA} &= \frac{f_m + m_f}{m_m + f_m + m_f + f_f}, \\
\text{naCoded} &= \frac{m_u + f_u}{m_m + f_m + m_f + f_f + m_u + f_u}, \\
\text{errorGenderBias} &= \frac{m_f - f_m}{m_m + f_m + m_f + f_f}.
\end{aligned}$$

Weighted Error

$$\text{weightedError}_w = \frac{\text{f}_\text{m} + \text{m}_\text{f} + w * (\text{m}_\text{u} + \text{f}_\text{u})}{\text{m}_\text{m} + \text{f}_\text{m} + \text{m}_\text{f} + \text{f}_\text{f} + w * (\text{m}_\text{u} + \text{f}_\text{u})}.$$

Distributions



Benchmark

Table 4 **Benchmark 1a: performance metrics for all services with their default gender assignments on the entire data set.** The *weightedError* is computed with $w = 0.2$.

	errorCoded	errorCodedWithoutNA	errorGenderBias	naCoded	weightedError
Gender API	0.0789	0.0503	−0.0111	0.0301	0.0562
gender-guesser	0.2224	0.0264	0.0022	0.2012	0.0731
genderize.io	0.1428	0.0502	0.0222	0.0974	0.0703
NameAPI	0.1794	0.0342	0.0037	0.1504	0.0672
NamSor	0.1282	0.0429	0.0072	0.0891	0.0613

Boxplot

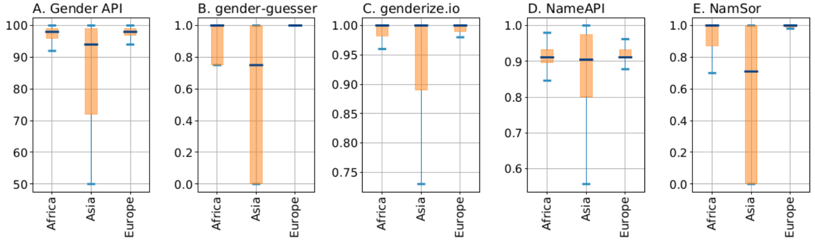


Figure 3 **Boxplots depicting quartiles for the confidence parameters of the gender inference services, split by geographical regions Africa, Asia, and Europe as returned by NamSor's origin API.** Panels (A), (B), (C), (D), and (E) display parameters *accuracy* of Gender API, self-constructed confidence of gender-guesser, *probability* of genderize.io, *confidence* of NameAPI and *scale* of NamSor, respectively. The bottom and top of the colored boxes mark the first and third quartiles of the distribution; the line in the middle of the boxes indicates the median; the ends of the whiskers correspond to the lowest and highest data points within 1.5 interquartile range.

Full-size DOI: 10.7717/peerjcs.156/fig-3

Boxplot, Name Origin

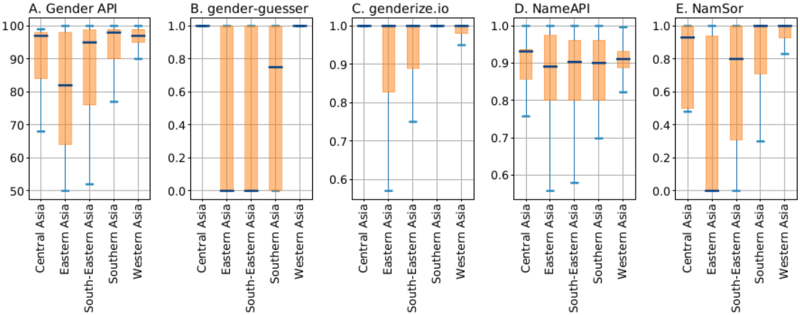


Figure 4 **Boxplots depicting quartiles for the confidence parameters of the gender inference services for Asian subregions as returned by NamSor's origin API, with boxplot settings as in Fig. 3.**

Full-size DOI: 10.7717/peerjcs.156/fig-4

Errors, Name Origin

Table 5 **Benchmark 1b, name origin: performance of all services with their default gender assignments in terms of the metrics *errorCoded* and *errorCodedWithoutNA*, broken down by name origin.** Values are rounded to four decimal figures.

	errorCoded			errorCodedWithoutNA		
	Africa	**Asia**	**Europe**	**Africa**	**Asia**	**Europe**
Gender API	0.0538	0.1759	0.0281	0.0469	0.112	0.0213
gender-guesser	0.2437	0.5171	0.0752	0.0365	0.0641	0.0147
genderize.io	0.1039	0.3282	0.0507	0.053	0.1206	0.0218
NameAPI	0.1505	0.3772	0.0807	0.0405	0.0897	0.0136
NamSor	0.0645	0.3459	0.0273	0.044	0.0903	0.0211

Damegender from the commands (I)

```
# Detect gender from a name
$ python3 main.py David
male
# Count gender from a csv example file
$ python3 csv2gender.py files/partial.csv
The number of males in files/partial.csv is 16
The number of females in files/partial.csv is 3
The number of gender not recognised in files/partial.csv is 2
# Count gender from a git repository
$ python3 git2gender.py https://github.com/chaoss/grimoirelab-p
The number of males sending commits is 15
The number of females sending commits is 7
```

Damegender from the commands (II)

```
# Count gender from a mailing list
$ cd files
$ wget -c http://mail-archives.apache.org/mod_mbox/httpd-annou
$ cd ..
$ python3 mail2gender.py http://mail-archives.apache.org/mod_mb
# Use an api to detect the gender
$ python3 api2gender.py David --api=genderguesser
male
# To measure success
$ python3 accuracy.py
Namsor accuracy: 0.9047619047619048
Sexmachine accuracy: 0.7619047619047619
```

Damegender from the commands (III)

```
$ python3 confusion.py
A confusion matrix C is such that Ci,j is equal to the number c
If the classifier is nice, the diagonal is high because there a
Namsor confusion matrix:
 [[ 3  0  0]
 [ 0 16  0]
 [ 0  2  0]]
Sexmachine confusion matrix:
 [[ 2  1  0]
 [ 2 14  0]
 [ 1  1  0]]

# To deploy a graph about correlation between variables
$ python3 corr.py
# To create the pickle models in files directory
$ python3 damemodels.py
```

Damegender as an exercise to practice NLTK and Perceval

```
~/git/python-examples/nlp/nltk: (dev) $ python3 sexmachine.py
What’s your name?: David
What’s my name?: Elena
David is male and Elena is female. Enjoy!.
The classifier has an accuracy: 0.052
Most Informative Features
             last_letter = ’a’              female : male   =
             last_letter = ’k’                male : female =
             last_letter = ’f’                male : female =
             last_letter = ’p’                male : female =
             last_letter = ’v’                male : female =
```

Damegender choosing features

```
$ python3 infofeatures.py
--------------------------------------------------------------
Females with last letter a: 0.4705246078961601
Males with last letter a: 0.048672566371681415
--------------------------------------------------------------
Females with last letter consonant: 0.2735841767750908
Males with last letter consonant: 0.6355328972681801
--------------------------------------------------------------
Females with last letter vocal: 0.7262612995441552
Males with last letter vocal: 0.3640823393612928
--------------------------------------------------------------
```

Damegender coding features

```
def features_int(self, name):
# features method created to check the scikit classifiers
    features_int = {}
    features_int["first_letter"] = ord(name[0].lower())
    features_int["last_letter"] = ord(name[-1].lower())
    for letter in 'abcdefghijklmnopqrstuvwxyz':
        features_int["count({})".format(letter)] = name.lower(
    features_int["vocals"] = 0
    for letter1 in 'aeiou':
        for letter2 in name:
            if (letter1 == letter2):
                features_int["vocals"] = features_int["vocals"]
    features_int["consonants"] = 0
    for letter1 in 'bcdfghjklmnpqrstvwxyz':
        for letter2 in name:
            if (letter1 == letter2):
                features_int["consonants"] = features_int["cons
```

Damegender the nltk standard model

```
def classifier(self):
    labeled_names = ([(name, 'male') for name in names.words('m
                    [(name, 'female') for name in names.words(
    featuresets = [(self.features(n), gender) for (n, gender) i
    train_set, test_set = featuresets[500:], featuresets[:500]
    classifier = nltk.NaiveBayesClassifier.train(train_set)
    return classifier

def guess(self, name, binary=False):
    guess = ''
    guess = super().guess(name, binary)
    if ((guess == 'unknown') | (guess == 2)):
        classifier = self.classifier()
        guess = classifier.classify(self.features(name))
        if binary:
            if (guess=='male'):
```

Damegender building a ML model (scikit)

```
def sgd(self):
# Scikit classifier
    X = np.array(self.features_list(path="files/all.csv"))
    y = self.gender_list("files/all.csv")
    clf = SGDClassifier(loss="log").fit(X,y)
    filename = 'files/sgd_model.sav'
    pickle.dump(clf, open(filename, 'wb'))
    return clf

def sgd_load(self):
    pkl_file = open('files/sgd_model.sav', 'rb')
    clf = pickle.load(pkl_file)
    pkl_file.close()
    return clf
```

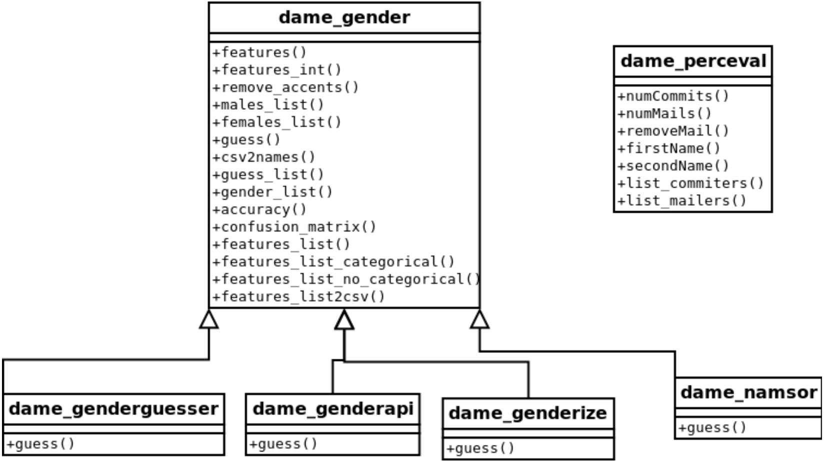
Damegender using a ML model (scikit)

```
$ cat main.py
```

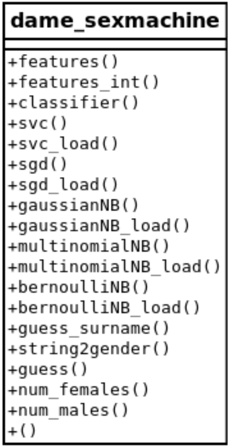
Damegender and perceval from string to gender

1. removeMail
2. string2array
3. string2gender (taking into account surnames and prefixes)

Damegender classes and methods (I)



Damegender classes and methods (II)



Damegender and the bussiness

- In CMS: wordpress, drupal, joomla
- In dictionaries: google translate, babylon, gnu dict, ...
- Enciclopedias: wikipedia, ...

–

- A good technical project has a good bussiness project and an interfaz for end users.
- A Free Software license and community can be a good point.