The Socface project, prediction of sex based on manuscript transcriptions

Martin Boutier
martin.boutier@ensae.fr
documented code at https://github.com/nabz78350/NLP

1 Introduction

The Socface project collaborates with archivists, demographers, and computer scientists to analyze and digitize French census data from 1836 to 1936, utilizing automatic handwriting recognition. Its goal is to compile a database from the handwritten census lists, organized by location and detailing each individual's name, birth year, and profession. This database aims to facilitate the study of social changes over a century and will be publicly accessible, offering free access to nominative lists of hundreds of millions of records. The project's objective is a binary classification task to predict the sex of individuals based on outputs from an automatic transcription engine. This approach presents challenges due to the inaccuracies and noise in the engine's predictions, emphasizing the need for robust supervised machine learning models to overcome the limitations of relying on ground truth data, which is not available under realistic conditions.

2 Data Analysis and Engineering

We have 232 samples for modeling, with a focus on predicting sex (0 for women, 1 for men) using features derived from transcriptions of original manuscript pictures, probably done by a computer vision model. The features include *Surname* (family name), *first name*, *Link* (to the person responsible for the sample), *Age* (at the time of sample collection), *birth date*, *lob* (location of birth), *civil status*, *occupation* (job or supporting activity), *employer* (name of employer or relationship to the person, e.g., Father), and *id* (mostly NaNs and unclear).

Corpus of French first names

We also have statistics about first names in France at our disposal. This table presents all names given within a certain period in France, along with the total number of males and females bearing these names. This is helpful as it allows us to infer the sex of the sample by examining the name and deducing its sex. By adopting this frequentist approach, we encode all first names with a binary classification: "prenom homme" if the frequency of this name being attributed to men is higher than 0.5, otherwise "prenom femme". Thus, we add a new feature called ***name sex***.

Sparsity

Our data exhibits significant sparsity. As indicated in [2], age, first name, surname, and possibly location of birth are the most reliable features with fewer NaNs. Due to the substantial sparsity and limited usefulness of some columns, the selected features for our analysis will be *first name*, *Link*, *birthdate*, *lob*, *occupation*, and *employer*.

Features and Dataset's Balance

Our data set is nearly balanced between men and women, with 46. 1% of the samples being women, as shown in the graph [4]. However, the relatively small sample size of 232 may lead to significant variability in test subsets. For example, a test subset could anomalously contain 80% males due to random variation. Although age is present in various groups, as illustrated in Figure [3], and is balanced, it is not considered a strong predictor of gender. We do not observe age / gender bias, such as a trend of women being younger than men. The predictive utility of some features is clear, particularly the *link* column, where specific keywords suggest gender. This is evident in both the pie chart of the *link* column in Figure [5] and the wordcloud in Figure [6]. For instance, the phrase "sa fille" (his daughter) implies that the sample is female. Conversely, the wordcloud of the employer column in Figure [7] shows this feature to be less informative.

Transcription Model Errors

A close examination of manuscript predictions reveals several errors. **These are crucial as they sometimes alter the real first name of a sample, preventing matching with a name in the French corpus of first names and thus excluding the**

sample from having a *name sex* value. This affects about 12% or 27 out of 232 samples. It is plausible that for samples with correct first name transcription, a simple binary classification model, such as Logistic Regression using only the name and its associated gender, could achieve near-perfect accuracy, barring small errors for gender-ambiguous names like *Camille* or *Claude*. Special attention is needed for cases where name transcription errors prevent reliance on the French corpus of first names, a likely scenario in real-life applications with more extensive data. **Throughout this work, we distinguish between *custom*, a split of the dataset in training and test where the test contains only cases with transcription errors (the "toxic" samples), and *not custom*, a random split using 33% of the sample at random for testing.** It is important to note that **we have never looked at the *groundtruth* column, as this data will not be available in a real-life scenario. We do not assess that a sample is toxic because its predicted first name does not match the groundtruth first name, but because its predicted first name does not match any first name in the corpus of French first names**.

3 Data Preprocessing

In addition to adding the feature *name sex* to our analysis, we implement several data preprocessing steps.

Data Augmentation via Random Cross-Sampling

Given the challenges posed by a small (232 samples), sparse, and error-prone dataset derived from transcription model predictions, data augmentation is used to mitigate overfitting and enhance model robustness for live deployment. The synthetic sample generation process includes:

1. Performing a train-test split, either with a custom test set of "toxic samples" or a pure random train-test split.
2. Separating the train dataset and the test.
3. Using the train dataset, collecting all values (including NaNs) for our features (*age*, *first name*, *surname*, *lob*, *link*, *occupation*, *employer*, *name sex*) separately for each sex class.
4. For each sex class, generating synthetic samples by randomly selecting values for each feature from within-class data.

This augmentation of the training set, **is performed after the train-test split to avoid test set contamination, and increases the training set size to 20,000 samples (10,000 per class), preserving feature sparsity to closely simulate the test set conditions.**

	before augmentation		After augmentation	
	custom split	not custom	custom split	not custom
train size	205	155	20 000	20 000
test size	27	77	27	77

Table 1: Comparison of Dataset Sizes Before and After Augmentation

Correcting Transcription Mistakes

Concerning the cases where transcription errors distort the first name (impacting 27 samples, referred to as toxic samples), making it impossible to link the sample's first name to the corpus of french first names, we explored several correction techniques.

1 - Using a BiLSTM Architecture

We developed a Bidirectional Long Short-Term Memory (BiLSTM) model to tackle the challenge of matching distorted name strings. The inputs are character embeddings of strings, truncated or padded to a maximum length of 8, formulated as a multiclass classification problem, targeting all names given more than 1000 times in the French first names corpus dataset (431 classes). The process involves:

- Selecting from the French corpus of first names all names given at least 250 times (431 names in total), serving as our labels.
- For these 431 names, performing 50 random alterations of the first name. These alterations include deleting or adding one or two random letters at random positions. For 50 iterations, each of the 4 operations is executed, creating 200 observations per original first name, totaling 172,400 labeled observations. An example of alterations for 4 first names is visible in appendix, see [4].
- Fitting a deep BiLSTM network, which uses the altered first names as inputs and the original first names as class labels, employing a softmax activation function and a multilabel cross-entropy loss function. Refer to [5] for the detailed architecture and [9] for the fitting results.
- When encountering a first name not matching any name in the French corpus (likely due to a transcription error), the BiLSTM model predicts the label of the first name and assigns the predicted first name to the sample.

The results of predicted first names for our *"toxic samples"* using the BiLSTM model are documented in Table [6]. The results are generally accurate, often recovering the exact original first name or, at the very least, the gender of the predicted first name matches the gender of the original first name, enhancing the chances for accurate gender classification.

2 - Using Python Package difflib

The **difflib.get_close_matches** function in Python identifies the best "close matches" to a string from a list of possibilities using the Ratcliff/Obershelp algorithm, which calculates the similarity ratio based on matching characters, with contiguous matches given higher weight.

3 - Using Fuzzy Matching

The **FuzzyWuzzy** package in Python uses the Levenshtein Distance to calculate differences between sequences, finding approximate matches and returning a score representing the similarity, with 100 indicating an exact match.

We benchmarked these three methods against scenarios where no correction is applied. No correction means that if a sample's first name does not match any name in the French corpus, it is left as it is, and we cannot infer the gender based on the first name, hence we cannot add the *sex name* feature : "prénom homme" or "prénom femme" to the features.

Merging features and creating proper string sequences

After the several steps of data preprocessing, we also treat stop words, by using french stop words from the Python package natural language toolkit (***nltk***). The last step is to merge all features for each sample to create one sequence. This will be our final and only training feature, called *message*. This is an instance of a sample (a man) which have been through the full data processing steps. A word cloud representation of the message sequences is visible in appendix, see[8].

message	*label*
cultinière chef prenom homme jean	1

4 Binary Classification Models

Predicting the sex of individuals based on textual messages involves binary classification, where the feature is the *message* and the target is binary (0 for woman, 1 for man). **This section explores a range of models, from simpler to more complex, for this task, including our deterministic benchmark**.

From String Sequence to Model Input Features

For natural language processing tasks, such as predicting sex from textual data, we convert string sequences into numerical inputs suitable for machine learning models through a two-step pipeline: 'CountVectorizer' and 'TfidfTransformer'. 'CountVectorizer' transforms text documents into a matrix of token counts by tokenizing the text into words or terms and counting their occurrences. This step numerically represents documents as vectors in a space where dimensions correspond to corpus tokens. Subsequently, 'TfidfTransformer' applies TF-IDF (Term Frequency-Inverse Document Frequency), a measure that evaluates a word's importance in a document relative to the corpus.

Benchmark

The benchmark we have designed is a straightforward deterministic binary classification rule for determining gender based on first names. **Specifically, the rule classifies a sample as male if the given first name is more commonly associated with males than females in a French corpus of first names. Conversely, it assigns a female classification if the name is more commonly associated with females. In the case where we do not use any correction method, for names not found in the corpus of first names, the model assigns a gender randomly**. This binary classification benchmark serves as a means to evaluate the performance of our transcription correction method. Essentially, if the correction algorithm suggests that a misspelled name is closer to a known male's first name, the benchmark will classify the sample as male, and similarly for female names. This approach allows us to assess the effectiveness of our name correction methodology in maintaining or improving classification accuracy.

Logistic Regression : Logistic regression offers a straightforward approach, prized for its simplicity and interpretability, ideal for linearly separable data. However, its linear nature may falter with the complex, non-linear patterns often present in text, necessitating regularization to prevent overfitting in high-dimensional spaces.

Naive Bayes Bernoulli Model : The Naive Bayes Bernoulli model, assuming feature independence, performs well in binary classification, even when this assumption is somewhat violated in textual data. It stands out for its simplicity and efficiency, but may struggle with datasets where feature interdependence is significant.

K-Nearest Neighbors Classifier : K-Nearest Neighbors (KNN) classifies based on the majority class among a sample's nearest neighbors, offering an intuitive and simple method. While it makes few assumptions about data distribution, its sensitivity to outliers and computational demands in large datasets, alongside the critical choice of k, are notable drawbacks.

XGBOOST Classifier: XGBOOST is effective for a variety of tasks, including text classification, due to its handling of sparse data and mechanisms to reduce overfitting. However, its complexity and the need for hyperparameter tuning make it less interpretable and more challenging to optimize than simpler models.

MLP Architecture Multilayer Perceptrons (MLP) are powerful, capable of modeling complex relationships through deep architecture and an embedding layer for text. Suggested configurations include two hidden layers with 120 units each. Despite their potential, MLPs risk overfitting and are computationally intensive with limited interpretability.

State-of-the-Art BERT Classification Model from Hugging Face: BERT, leveraging transformer architecture, represents the forefront of natural language processing, offering exceptional performance in text classification. Although it requires significant computational resources and presents challenges in interpretability, fine-tuning BERT for our dataset promises high accuracy.

For sklearn models (Logistic Regression, Naive Bayes Bernoulli Model, KNN, and XGBOOST), **a grid search has been conducted (5 folds), and the parameter grid is detailed in appendix, see [2]. For the MLP model, the parameters are fixed, also detailed in the appendix, see [5]**.

5 Results

Our analysis focuses on two primary objectives:

1. Evaluating the efficacy of various methods for correcting misspellings in first names caused by the computer vision transcription model. These methods include difflib, Fuzzy, and BiLSTM.
2. Assessing the performance of the binary classification machine learning model.

To accurately interpret our findings, it is essential to understand the following key points:

- **Custom Split:** A setting where *Custom* = *True* indicates that the division of data into training and testing sets is structured so that all observations with transcription errors in the first name (toxic samples) are allocated to the test set. This scenario represents a potential real-life challenge, offering insights into the model's performance under adverse conditions. Conversely, a non-custom split (i.e., *Custom* = *False*) implies a random allocation of data into training and testing sets, allocating 33% of the dataset to the test. In both approaches, the training dataset is then augmented through random cross-sampling before fitting the model, which is then evaluated on the test dataset.
- **Correction Method:** This term specifies the technique employed to rectify misspelled names to match an existing name within the French corpus of names. Options include *None* (where no correction is applied), *difflib*, *Fuzzy*, and *BiLSTM*.

The delineation between a custom and a non-custom split is crucial, as it reflects different testing scenarios that could influence the interpretation of the model's effectiveness. Furthermore, understanding the role and type of correction method applied is vital for comprehending how misspellings are addressed by our correction methods and their impact on the overall results.

First, regarding the benchmark, one can observe the effectiveness of error correction methods from the transcription models. Since the first name is the top predictor for gender, we find that the BiLSTM correction method yields the best results on the benchmark and a custom split. With a 77% accuracy on the test set using BiLSTM, this means that for the toxic samples where the first name is poorly transcribed, the BiLSTM model assigns a first name from the corpus that corresponds to the correct gender in 77% of cases, if not the original name itself. This is much better than without correction, where we have a test accuracy of 51%; this corresponds to the mere chance of our benchmark when the name is poorly transcribed and the benchmark randomly chooses the gender classification.

Overall, our results are very similar across models; it seems that we do not need to leverage complex models, which is logical given that there is not a particular sense of attention in our text sequence, nor is there importance for the order of words in the sequence. Our data augmentation via random cross-sampling nevertheless allowed us to conduct a proper grid search cross-validation to train our models, as well as to train complex models like the MLP or the BertForSequenceClassification as well, respecting the rule of 8 to 20 tokens per parameter. A significant achievement is the method we employ to try to correct the prediction errors of the transcription model. Not at a single step, did we use the ground truth data, which would not be accessible in real life. Each method of correction enhances results, especially when we use a test set made of all samples that have a first name that doesn't match any name in the corpus of names (what we called a custom split). As we can see in [10], using a correction method systematically improves our results in terms of test accuracy, especially when doing a custom test split, which is intuitive.

In terms of analyzing the results, the BiLSTM method we fitted seems to yield better results, especially on a custom split where it systematically outperforms the results when no correction is added, and this for every classification model fitted afterward.

In terms of models performance, as seen in the plot [1] or the table in the appendix [5], there are no clear winners at first sight. We can eliminate the XGBOOST model, which is too unstable and sensitive to the train-test split. Similarly, the MLP model is subject to instability in training, as seen in [**??**], showcasing potentially too much complexity or a wrong choice of architecture. On a custom split, meaning all samples with no matching first names in the corpus of first names are in the test set, the Logistic Regression and the Bert model achieve good results with a BiLSTM correction method, both yielding an 85.20% accuracy. Conducting a custom train-test split this way simulates what would be a worst-case scenario, e.g., a stress-test, in the case where we have to predict samples where each first name does not match any known French first name. By proceeding with a pure

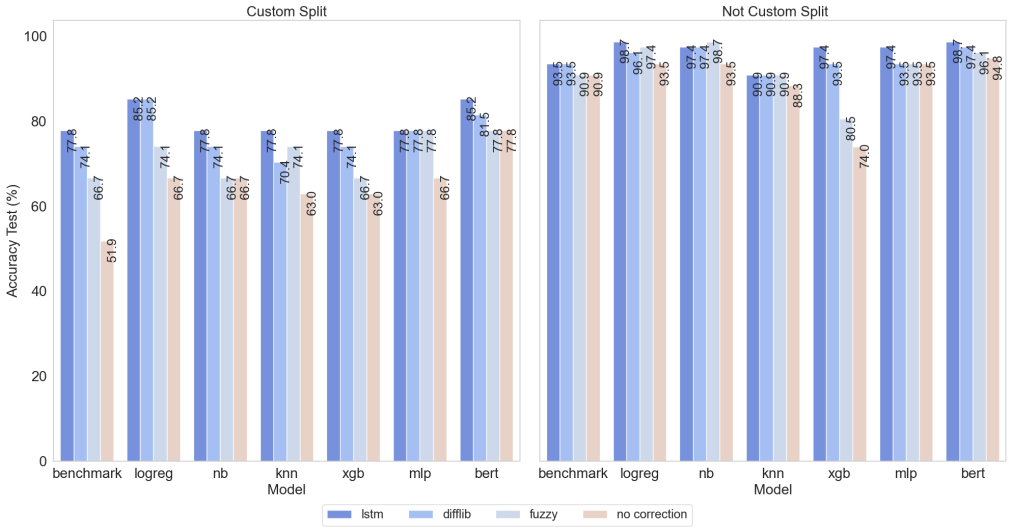

Figure 1: Test accuracy across different correction methods, including doing no correction, and different test-train split (custom or not custom)

random train-test split with 33% of the original 232 samples in the test, we can still see that using a correction method improves our results, but less than in a custom test split. This is very logical as some "toxic" samples might still be randomly in the test set, and our correction method allows predicting them with greater accuracy, especially the BiLSTM correction method, which always systematically yields slightly better results than using *difflib* or *FuzzyWuzzy*. In randomly splitting into train and test, Bert or the Logistic Regression both coupled with a BiLSTM correction method, yields a 98.7% accuracy on the test set.

Conclusion

In such a problem, we've seen that the final classification models has less importance than the treatment we do to process textual data. Generally, yes, Bert is yielding the best results, both for a Custom test split and a random test split, but it is a very complex, hardly interpretable model, not speaking about the costly computation time. The Logistic Regression, or the Naive Bayes Bernoulli Model, gives very similar performance, while being much more simple and accessible, which is always better in production. What is certain is that it is crucial to detect when the models that predict manuscript images makes mistakes, so that we can try to correct them and predict the sex with greater accuracy. We were able to detect these mistakes without looking at the ground truth column, just by looking when a "VLOOKUP" with the corpus of French first names at our disposal didn't produce any matching name, meaning that either the first name of this simple is a very rare one, or more likely that there is an error in retransmitting the first name. At this extent, our BiLSTM correction method yields better results than plug-and-play Python solutions *difflib* or *FuzzyWuzzy*. Overall, a simple model such as Logistic Regression, backed by BiLSTM misspelling correction model, can yield very good results, both in a test set full of toxic samples (85.2% accuracy) and in a random train-test split (98.7% accuracy). We can also argue that the BertSequenceClassification model achieves exactly the same results, but if one follows Occam's razor principle, the Logistic Regression is the way to go.

Appendix

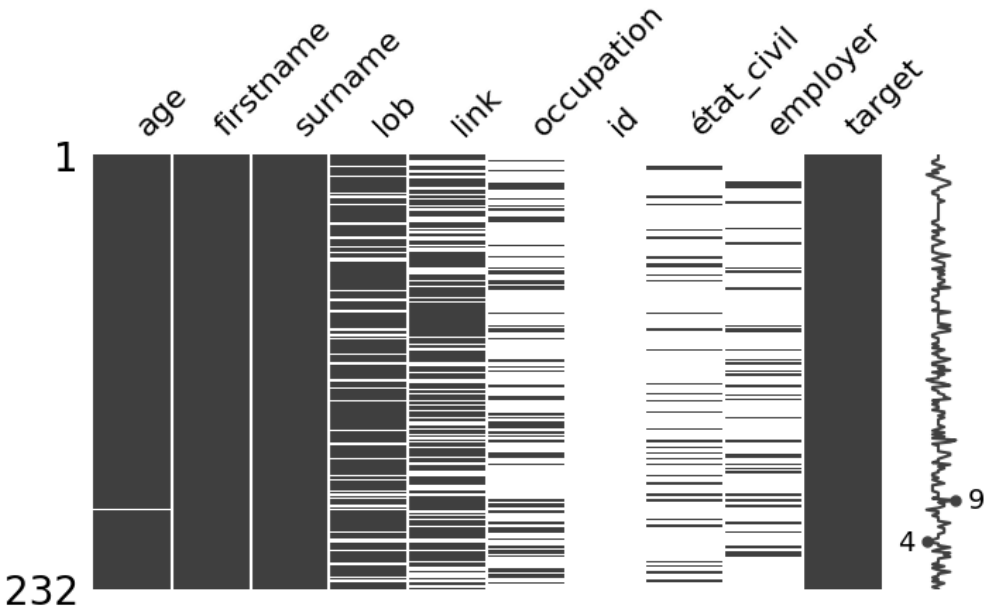

Figure 2: NaN representation in the dataset

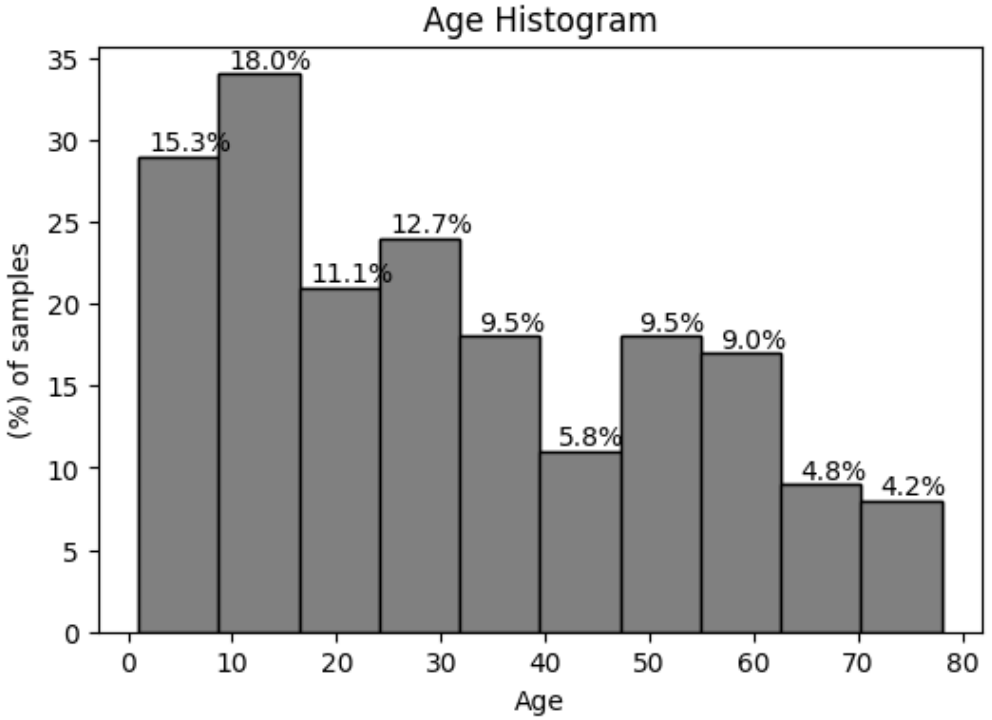

Figure 3: Histogram of Age

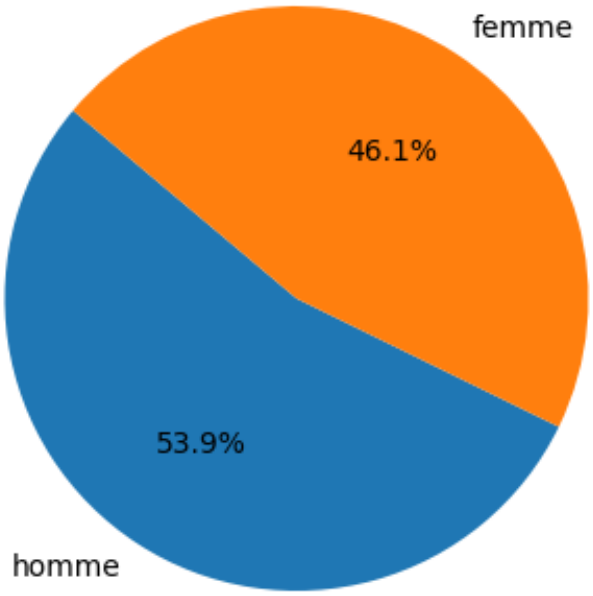

Figure 4: Sex Reparition

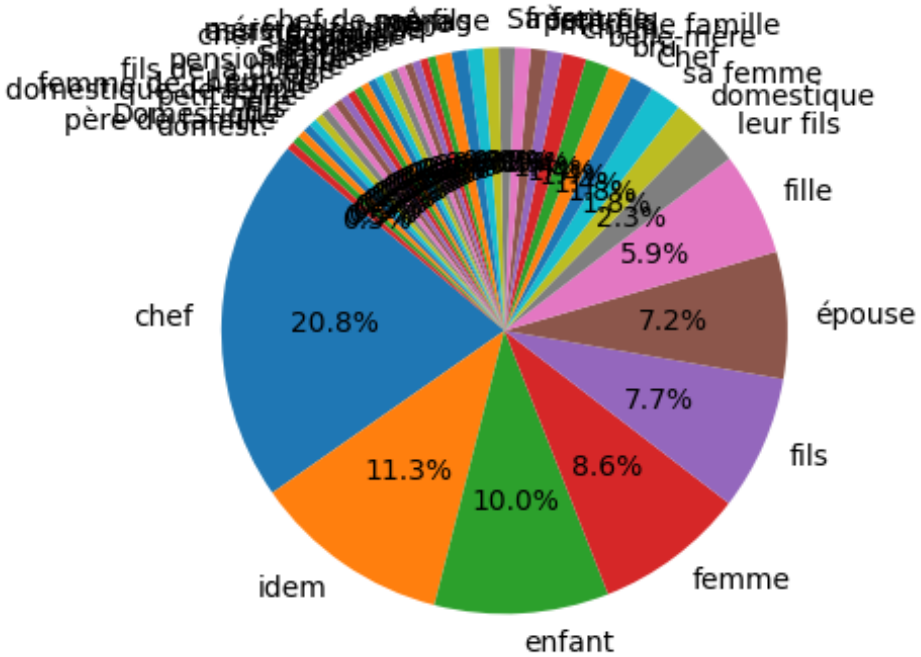

Figure 5: Pie Chart of the "link" column

Figure 6: Wordcloud representation of the "link" column

Figure 7: Wordcloud representation of the "employer" column

Figure 8: WordCloud plot of the sequences used as inputs for models

Model	**Parameters**
KNeighborsClassifier	n_neighbors: [3, 5, 10], metric: ['euclidean', 'manhattan']
BernoulliNB	alpha: [0.5, 1.0, 2.0], binarize: [0.0, 1.0]
LogisticRegression	C: [0.01, 0.1, 1, 10], penalty: ['l1', 'l2']
XGBClassifier	learning_rate: [0.01, 0.1, 0.2], max_depth: [3, 5, 7], n_estimators: [100, 200]

Table 2: Grid Search Parameters for sklearn-based models

Layer (type)	Output Shape	Param #
Embedding (Embedding)	(None, 100, 500)	250 000
Flatten (Flatten)	(None, 50000)	0
Dense (Dense)	(None, 100)	5 000 100
Dropout (Dropout)	(None, 100)	0
Dense_1 (Dense)	(None, 100)	10 100
Dropout_1 (Dropout)	(None, 100)	0
Dense_2 (Dense)	(None, 100)	10 100
Dropout_2 (Dropout)	(None, 100)	0
Dense_3 (Dense)	(None, 1)	101
Total params:		5 270 401
Trainable params:		5 270 401
Non-trainable params:		0

Table 3: MLP architecture, fitted on 100 epochs, with a callback patience on validation loss of 10 epochs. Batch size used is 32, learning rate 3e-4, sigmoid activation function and a binary cross entropy loss for a 0-1 classification task

Original First Name	jean	madeleine	marie	louis	jeanne
Alteration 1	jpau	madblsine	merie	ligis	jeangd
Alteration 2	cean	modeleine	mazie	louzs	joanne
Alteration 3	jen	madeline	arie	luis	jenne
Alteration 4	ja	mdeline	are	los	jenze

Table 4: Visualization of Original First Name (model labels) and Altered First Name (model inputs) for 5 random names in the corpus of French First Names.

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 8, 32)	16,000
dropout (Dropout)	(None, 8, 32)	0
bidirectional (Bidirectional)	(None, 8, 256)	164,864
batch_normalization (Batch Normalization)	(None, 8, 256)	1,024
dropout_1 (Dropout)	(None, 8, 256)	0
bidirectional_1 (Bidirectional)	(None, 256)	394,240
batch_normalization_1 (Batch Normalization)	(None, 256)	1,024
dense (Dense)	(None, 128)	32,896
dropout_2 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 1005)	129,645
Total params: 739,693		
Trainable params: 738,669		
Non-trainable params: 1,024		

Table 5: BiLSTM Architecture, fitted on 100 epochs, with a callback of 10 epochs patience on validation loss, with a validation split of 25 %

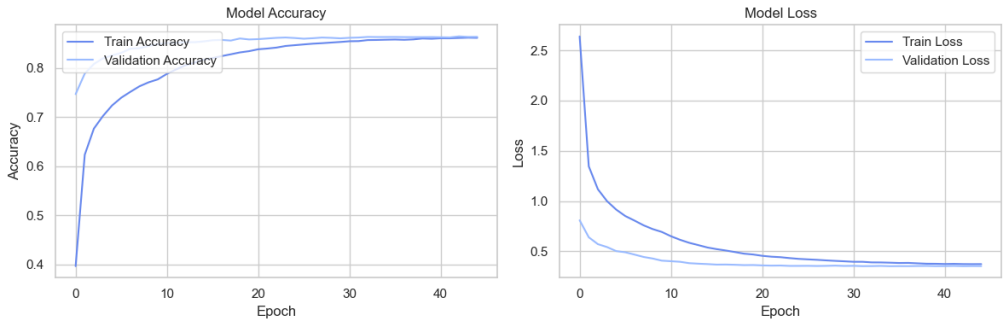

Figure 9: Training fit history for the BiLSTM firstname correction model

	Firstname Groundtruth	Transcribed Firstname	Corrected Firstname with BiLSTM	**Inferred Sex Class Feature**
18	Antoine	Angène	angèle	prénom femme
24	Alexandre	Herandre	alexandre	prénom homme
25	Marguerite	Oarguerite	marguerite	prénom femme
34	Virginie	Vigmie	virginie	prénom femme
65	Antoine	Angloise	angeline	prénom femme
69	François	Feris	felix	prénom homme
82	Jacques	Jregues	jacques	prénom homme
104	Gilbert	Gilbeuse	gilbert	prénom homme
121	Branislav	Branistau	francisque	prénom homme
126	Stanislas	Mamiolas	mathias	prénom homme
132	Madeleine	Gadeleine	madeleine	prénom femme
138	Etiennette	Ctiennette	etiennette	prénom femme
142	Antoine	Aupène	eugène	prénom homme
147	Claude	Clause	claude	prénom homme
159	Gilbert	Gilbeup	gilbert	prénom homme
193	Marthe	Marthy	marthe	prénom femme
197	Françoise	Faul	paul	prénom homme
200	François	Zean	jean	prénom homme
216	Victor	Vicher	victor	prénom homme
217	Madeleine	Mareleine	madeleine	prénom femme
219	France	Franco	francois	prénom homme
225	Blaise	Clause	claude	prénom homme
231	Madeleine	Vadeleine	madeleine	prénom femme
232	Théodore	Théodote	théodore	prénom homme
233	Barthélémy	Barthe	marthe	prénom femme
234	François	Ferd	fernand	prénom homme
236	Pétronille	Gihromille	pétronille	prénom femme

Table 6: All toxic samples and the predicted first name with the BiLSTM correction model. Color code correspond to green when the BiLSTM predicted firstname correspond to a sex that matches the sex of the groundtruth first name (using frequency of sex and given firstnames), even if the BiLSTM prediction is not exactly the same as the groundtruth first name.

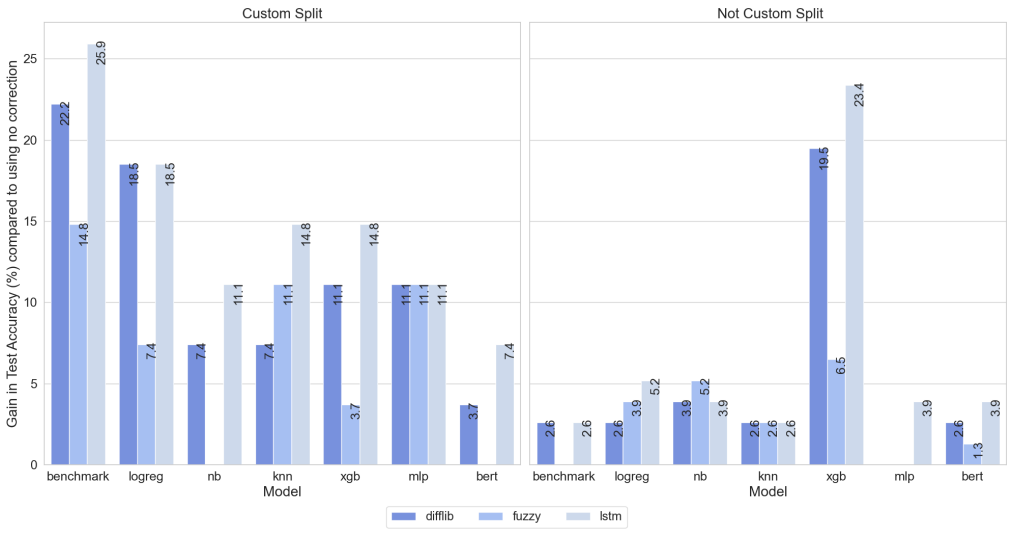

Figure 10: Accuracy Test improvements when using a correction method, compared to using no correction method

Model	custom split	correction method	Train accuracy (%)	Test accuracy (%)
benchmark	custom split	difflib	98.536585	74.074074
		fuzzy	98.536585	66.666667
		lstm	98.536585	77.777778
		no correction	98.536585	51.851852
	not custom split	difflib	94.193548	93.506494
		fuzzy	94.193548	90.909091
		lstm	94.838710	93.506494
		no correction	94.193548	90.909091
logreg	custom split	difflib	100.000000	85.185185
		fuzzy	98.536585	74.074074
		lstm	95.609756	85.185185
		no correction	98.536585	66.666667
	not custom split	difflib	96.774194	96.103896
		fuzzy	95.483871	97.402597
		lstm	94.838710	98.701299
		no correction	98.709677	93.506494
nb	custom split	difflib	98.536585	74.074074
		fuzzy	98.536585	66.666667
		lstm	98.536585	77.777778
		no correction	98.536585	66.666667
	not custom split	difflib	97.419355	97.402597
		fuzzy	96.774194	98.701299
		lstm	96.774194	97.402597
		no correction	100.000000	93.506494
knn	custom split	difflib	95.609756	70.370370
		fuzzy	98.536585	74.074074
		lstm	98.536585	77.777778
		no correction	95.609756	62.962963
	not custom split	difflib	94.838710	90.909091
		fuzzy	94.838710	90.909091
		lstm	94.838710	90.909091
		no correction	96.129032	88.311688
xgb	custom split	difflib	98.536585	74.074074
		fuzzy	98.536585	66.666667
		lstm	98.536585	77.777778
		no correction	98.536585	62.962963
	not custom split	difflib	100.000000	93.506494
		fuzzy	100.000000	80.519481
		lstm	100.000000	97.402597
		no correction	100.000000	74.025974
mlp	custom split	difflib	100.000000	77.777779
		fuzzy	100.000000	77.777779
		lstm	100.000000	77.777779
		no correction	99.024391	66.666669
	not custom split	difflib	99.354839	93.506491
		fuzzy	98.709679	93.506491
		lstm	99.354839	97.402596
		no correction	98.709679	93.506491
bert	custom split	difflib	100.000000	81.481481
		fuzzy	100.000000	77.777778
		lstm	100.000000	85.185185
		no correction	100.000000	77.777778
	not custom split	difflib	96.129032	97.402597
		fuzzy	96.129032	96.103896
		lstm	95.483871	98.701299
		no correction	94.193548	94.805195

Table 7: Table of results for all models, method of splitting, and mispelled firstname correction method