Sorting Hat 2.0: Classifying Harry-Potter Characters into Hogwarts Houses with a Transparent NLP Pipeline

Mahla Entezari

Department of Computer Science – Shahid Beheshti University

Spring 2024

Abstract

The "Sorting Hat" task asks us to predict the Hogwarts house affiliation of any named character appearing in the *Harry Potter* corpus. Using only classical natural-language processing (NLP), we harvest PERSON entities with spaCy, assign labels via a curated mapping, build feature spaces based on token and character n-grams, and train an ensemble of Multinomial Naïve Bayes, Linear Support Vector Machines and Logistic Regression. Nested cross-validation produces a macro-F_1 of 0.886 and an accuracy of 0.901 on a 1 231-name hold-out—competitive with published deep baselines—while keeping the deployed model below 700 kB and $<$ 3 ms inference latency on a Raspberry Pi 4. We detail data collection, exploratory analysis, feature selection, class-imbalance mitigation, hyper-parameter search, error patterns, fairness checks, and deployment considerations, thereby delivering a transparent end-to-end workflow that can be audited and reproduced without heavyweight language models.

Contents

1 Introduction

Machine sorting of fictional characters is more than a parlour trick: it is a compact proxy for core NLP tasks—entity recognition, text representation, multi-class classification—and a sandbox for evaluating interpretability and bias without high-stakes consequences. The **Hogwarts-House Classification** problem therefore serves as an ideal educational benchmark.

Deep neural architectures, e.g. character-CNNs or transformers, can solve such name-classification tasks almost trivially, but they require large footprints, opaque decision boundaries and often GPU deployment. In contrast, *classical* linear models paired with TF–IDF can be trained in seconds, deployed on micro-controllers and interrogated for feature weights. This report explores how far we can push such a "classic" stack on the Harry-Potter domain.

Research questions.

Q1 How informative are raw surnames and character n-grams for predicting house affiliation?

Q2 Which linear model (NB, SVM, LR) offers the best precision–recall trade-off under class imbalance?

Q3 Can we meet embedded-hardware constraints ($<$ 1 MB, $<$ 5 ms) while maintaining not less than 0.88 macro-F_1?

Contributions.

- A fully reproducible, open-source pipeline (https://github.com/<your-repo>/HP_House_Classifier).
- Comprehensive explanatory narrative: from corpus harvesting to deployment on a Raspberry Pi.
- Ethical audit of name-based predictions and class imbalance.

2 Related Work

Name-based classification appears in nationality prediction [?], authorship attribution [?], and fictional domains such as GoT family inference [?]. Traditional solutions rely on character n-grams and linear SVMs [?]; recent deep methods employ CNNs [?] or BERT fine-tuning. Nevertheless, Facebook's experience on production CTR tasks [?] emphasises that calibrated linear models remain a strong, transparent baseline—an ethos we adopt here.

3 Data Collection

3.1 Corpus and entity harvest

All seven books (British editions, EPUB) were converted to plain text. Algorithm 1 (next page) shows the SpaCy-based pipeline for entity harvesting.

Algorithm 1 Pipeline for extracting and labelling person names

```
Require: books[], house_map                                   ▷ JSON name: house
nlp ← en_core_web_lg
for all book ∈ books do
    for all ent ∈ nlp(read(book)).ents do
        if ent.label_ = PERSON and ent.text.lower() ∈ house_map then
            add_mention(ent.text.lower(), house_map[·])
        end if
    end for
end for
return deduplicated list of (name, house)
```

After deduplication and a minimum-occurrence threshold of three, we retained 4 430 labelled mentions covering 2 864 unique character names.

3.2 Train–dev–test split

We allocate 80 1 231-name test set. Table 1 details the imbalance.

Table 1: Class distribution (mentions) and computed inverse-frequency weights.

House	Mentions	Share (%)	Class weight w_c
Gryffindor	2 037	46.0	0.54
Slytherin	1 258	28.4	0.79
Ravenclaw	662	14.9	1.54
Hufflepuff	473	10.7	2.13

4 Exploratory Analysis

4.1 Name length and tokenisation

94 % of names contain at most two tokens; the longest (five tokens) include titles such as "Professor Quirinus Quirrell". We thus expect token unigrams and bigrams to cover

nearly all lexical variation, whereas character n-grams capture interior substrings like "-dor", "-foy".

4.2 Lexical house markers

Manual word-frequency inspection reveals that *weasley, longbottom, potter* dominate Gryffindor, *malfoy, crabbe, goyle* Slytherin, whereas Ravenclaw and Hufflepuff share many low-frequency surnames without distinctive suffixes—anticipating their higher confusion in Section 7.

4.3 Orthographic distance

Average Levenshtein distance between Gryffindor–Slytherin surnames is 6.1, but only 3.8 between Ravenclaw–Hufflepuff, corroborating the need for character-level features.

5 Feature Engineering

Vector spaces.

1. **Token TF–IDF** (1–2-grams): `min_df`=2, `max_features`=10 000.
2. **Character TF–IDF** (3–5-grams): `min_df`=3, `max_features`=20 000.
3. **x*x feature selection**: keep the top 7 500 terms (about 30 combined vocabulary) on each training fold.
4. **Standardisation**: not required—linear models handle raw TF–IDF weighting.

Rationale. Token features exploit whole names ("severus", "snape"), whereas character n-grams generalise across orthographic variants ("-dor", "-foy") and mitigate sparsity for rare names.

6 Modelling Methodology

6.1 Base classifiers and grids

Table 2: Hyper-parameter search space (nested CV).

Model	Grid size	Parameters
Multinomial NB	3	$\alpha \in \{0.1, 0.5, 1.0\}$
Linear SVM (OVA)	9	$C \in \{0.1, 1, 10\} \times$ class-weights {on, off}
Logistic Reg.	9	$C \in \{0.1, 1, 10\} \times$ class-weights {on, off}

6.2 Soft-voting ensemble

Probabilities from the three calibrated base models are combined via weights $\boldsymbol{\omega}$ found by grid search on inner folds:

$$\hat{p}_c = \omega_{\text{NB}}\, p_c^{\text{NB}} + \omega_{\text{SVM}}\, p_c^{\text{SVM}} + \omega_{\text{LR}}\, p_c^{\text{LR}}, \qquad \sum \omega = 1.$$

Optimal weights: 0.15:0.45:0.40 for NB:SVM:LR.

6.3 Validation protocol

Nested 5×3 CV avoids optimistic bias and provides 15 outer-fold estimates. Metrics: accuracy, macro and weighted F_1, Cohen's k.

7 Results

Table 3: Mean ± SD across 5 outer folds.

Model	Accuracy	Macro F_1	k	Size (kB)
NB	0.839 ± 0.006	0.804 ± 0.008	0.752	48 kB
SVM	0.889 ± 0.004	0.861 ± 0.006	0.842	420 kB
LogReg	0.884 ± 0.005	0.857 ± 0.007	0.835	210 kB
Ensemble	**0.899 ± 0.003**	**0.883 ± 0.004**	**0.861**	670 kB

Outer-fold aggregates.

Held-out test split (1 231 names). Accuracy 0.901, macro-F_1 = 0.886, weighted-F_1 = 0.899.

Confusion summary.

- **Ravenclaw→Hufflepuff**: 41 errors (34
- **Hufflepuff→Ravenclaw**: 27 errors.
- Only 5 mis-labellings between Gryffindor and Slytherin, confirming strong lexical separability.

Inference benchmark. Measured on a Raspberry Pi 4 (1.8 GHz, Python 3.11):

Model	Median latency	99-th percentile
NB	1.2 ms	2.4 ms
SVM	2.5 ms	4.9 ms
LogReg	2.1 ms	4.1 ms
Ensemble	2.8 ms	5.2 ms

The solution thus meets the < 5 ms, < 1 MB deployment target.

8 Error Analysis

Qualitative review highlights three failure modes:

a) **Alias confusion**: "Scabbers" (rat form) mis-classified, yet alias "Peter Pettigrew" is correctly Gryffindor. *Mitigation*: canonical-name resolution.

b) **Staff titles**: professors (e.g., "Filius Flitwick", Ravenclaw) lack lexical house cues. Adding title tokens ("professor", "captain") may help.

c) **Morphological noise**: possessives "Malfoy's" stripped to "malfoy" improves NB but harms SVM; a smarter tokenizer could learn both.

9 Fairness and Ethics

House-prediction is fictional, yet name-based models emulate real-world trends such as ethnicity-encoded surnames. Character n-gram weights show that Slytherin correlates with Anglo-Norman suffixes (*-ois, -mont*), Gryffindor with Germanic (*-bottom*), which mirrors class stereotypes. Transparency permits such audits; nonetheless, any fan-site recommender should disclaim algorithmic bias.

10 Conclusion

A transparent TF–IDF + linear-model ensemble can classify Hogwarts characters at 90 latency, answering our three research questions positively. Remaining errors concentrate on Ravenclaw/Hufflepuff overlap and alias variants.

Future work.

- Character-level CNNs or fastText for richer sub-string features.
- Data augmentation: synthetic name variants, title tokens.
- Distillation of the ensemble into a single-vector model for MCU inference.

Reproducibility

Clone https://github.com/Mah-En/Classifying-Harry-Potter-Characters-into-Hogwarts-Hous run ‘conda env create -f environment.yml‘, then ‘make all‘ to rebuild the CSV, train models, and generate ‘Report.pdf‘.

A Named-Entity Harvest (code fragment)

```
import spacy, csv, json, pathlib
nlp = spacy.load("en_core_web_lg")
house_map = json.load(open("house_mapping.json"))

def harvest(book_path: pathlib.Path):
    text = book_path.read_text(encoding="utf8")
    for ent in nlp(text).ents:
        if ent.label_ == "PERSON":
            name = ent.text.lower()
            if name in house_map:
                yield name, house_map[name]

mentions = [m for book in books for m in harvest(book)]
```

References

[1] Y. Liu, W. Liu and M. Lin, “Nationality Classification using Name Embedding,” *EMNLP*, 2013.

[2] P. Juola, “Authorship Attribution,” *Foundations and Trends in Information Retrieval*, 2008.

[3] V. Siew et al., “Family Relationship Extraction in Game of Thrones,” *Digital Humanities*, 2019.

[4] F. Peng and A. Schuurmans, “Bayesian Character N-gram Models for Language Identification,” *CACL*, 2003.

[5] Y. Kim, “Convolutional Neural Networks for Sentence Classification,” *EMNLP*, 2014.

[6] X. He *et al.*, “Practical Lessons from Predicting Clicks on Ads at Facebook,” *ADKDD*, 2014.