Statistical learning with high-cardinality string categorical variables

Patricio Cerda Reyes

université PARIS-SACLAY

Inria INVENTEURS DU MONDE NUMÉRIQUE

UNIVERSITÉ PARIS SUD
Comprendre le monde, construire l'avenir

NNT : 2019SACLS470

Thèse de doctorat

Apprentissage statistique à partir de variables catégorielles non-uniformisées

Thèse de doctorat de l'Université Paris-Saclay
préparée à l'Université Paris-Sud à l'INRIA

Ecole doctorale n°580 Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Palaiseau, le 28 Novembre 2019, par

PATRICIO CERDA REYES

Composition du Jury :

Marc Schoenauer Directeur de recherche, Inria (équipe TAU)	Président
Laurent Charlin Professeur adjoint, HEC Montréal	Rapporteur
Stéphane Gaïffas Professeur, Université Paris Diderot (LPSM)	Rapporteur
Charles Bouveyron Professeur, Université Côte d'Azur	Examinateur
Patrick Valduriez Directeur de recherche, Inria (LIRMM)	Examinateur
Balázs Kégl Directeur de recherche, Huawei / CNRS	Examinateur
Gaël Varoquaux Directeur de recherche, Inria (équipe Parietal)	Directeur de thèse

STATISTICAL LEARNING WITH HIGH-CARDINALITY STRING CATEGORICAL VARIABLES

PATRICIO CERDA REYES

Dissertation submitted in partial fulfillement of the requirements for the degree of Doctor of Philosophy.

Université Paris-Saclay
Sciences et Technologies de l'Information et de la Communication

October 2016 – September 2019
Inria Saclay. Palaiseau, France

PH.D. COMMITTEE

PRESIDENT:
Pr. Marc Schoenauer, Inria (team TAU), France.

DIRECTOR:
Pr. Gaël Varoquaux, Inria (team Parietal), Palaiseau, France.

REVIEWERS:
Pr. Laurent Charlin, HEC Montréal, Montréal, Canada.
Pr. Stéphane Gaïffas, Université Paris Diderot (LPSM), Paris, France.

EXAMINERS:
Pr. Charles Bouveyron, Université Côte d'Azur, Nice, France.
Pr. Patrick Valduriez, Inria (LIRMM), Montpellier, France.
Pr. Balázs Kégl, Huawei/CNRS, France.

This thesis was prepared at Inria Saclay (team Parietal), from October 2016 to September 2019. It was funded by the Wendelin and Dirty-Data (ANR-17-CE23-0018) grants.

ABSTRACT

Tabular data often contain columns with *categorical variables*, usually considered as non-numerical entries with a fixed and limited number of unique elements or *categories*. As many statistical learning algorithms require numerical representations of features, an encoding step is necessary to transform categorical entries into feature vectors, using for instance *one-hot encoding*. This and other similar strategies work well, in terms of prediction performance and interpretability, in standard statistical analysis when the number of categories is small.

However, non-curated data give rise to string categorical variables with a very high cardinality and redundancy: the string entries share semantic and/or morphological information, and several entries can reflect the same entity. Without any data cleaning or feature engineering step, common encoding methods break down, as they tend to lose information in their vectorial representation. Also, they can create high-dimensional feature vectors, which prevent their usage in large scale settings.

In this work, we study a series of categorical encodings that remove the need for preprocessing steps on high-cardinality string categorical variables. An ideal encoder should be: scalable to many observations; interpretable to end users; and capture the morphological information contained in the string entries.

Experiments on real and simulated data show that the methods we propose improve supervised learning, are adapted to large-scale settings, and, in some cases, create feature vectors that are easily interpretable. Hence, they can be applied in Automated Machine Learning (AutoML) pipelines in the original string entries without any human intervention.

RÉSUMÉ

Les données de type tabulaire contiennent souvent des *variables catégorielles* : des colonnes non numériques avec un nombre fixe et limité d'éléments uniques, appelés *catégories*. De nombreux algorithmes d'apprentissage statistique nécessitent une représentation numérique des variables catégorielles. Une étape d'encodage est donc nécessaire pour transformer ces entrées en vecteurs. Pour cela, plusieurs stratégies existent, dont la plus courante est celle de *l'encodage one-hot*. Cette dernière fonctionne bien dans le cadre de l'analyse statistique classique (sur le plan de la puissance de prédiction et de l'interprétation) lorsque le nombre de catégories reste faible.

Cependant, les données catégorielles non-uniformisées présentent le risque d'avoir une grande cardinalité et des redondances. En effet, les entrées peuvent partager des informations sémantiques et/ou morphologiques, et par conséquent, plusieurs entrées peuvent refléter la même entité. Sans une étape de nettoyage ou d'agrégation au préalable, les méthodes d'encodage courantes peuvent perdre en efficacité du fait d'une représentation vectorielle erronée. En outre, en présence de redondances, le risque d'obtenir des vecteurs de très grande dimensionnalité croît avec la quantité de données, ce qui empêche leur utilisation dans l'analyse statistique de données volumineuses. En général, les analystes de données s'attaquent à ce problème avec des techniques de nettoyage des données ou de *feature engineering*. L'objectif étant de créer de nouvelles variables dépendantes pour faciliter l'apprentissage statistique. Cependant, ces techniques manuelles sont chronophages et nécessitent parfois un certain niveau d'expertise dans le domaine spécifique à la base de donnée préalable.

Dans cette thèse, nous nous concentrons sur la recherche de représentations numériques pour les variables catégorielles à cardinalité élevée qui font partie de l'ensemble des variables dépendantes. Nous étudions une série de méthodes d'encodage qui permettent de travailler directement sur des variables catégorielles à grande cardinalité, sans qu'il soit nécessaire de les traiter en amont. Notre objectif est de fournir des vecteurs pour les données catégorielles qui : *(i)* sont de dimensionnalité limitée ; *(ii)* sont scalables ; qui *(iii)* améliorent les performances de l'analyse statistique telles que l'apprentissage supervisé ; et qui font tout cela *(iv)* sans aucune intervention humaine. De plus, comme de nombreuses applications d'apprentissage automatique nécessitent des algorithmes explicables par l'homme, nous étudions aussi l'interprétabilité de ces encodeurs.

Pour évaluer ces nouvelles méthodes d'encodage, nous avons collecté 17 jeux de données contenant au moins une variable catégorielle à cardinalité élevée. A chacun de ces jeux de donnés, nous avons associé une tâche de prédiction spécifique. À notre connaissance, il s'agit de la première collection de données liés à ce sujet. Tous les jeux de données sont open source et nous espérons qu'ils favoriseront davan-

tage des futures recherches dans ce domaine. Une première contribution importante de cette thèse est la généralisation de l'encodage one-hot, appelé *similarity encoding*, qui prend en compte la similarité entre deux chaînes de caractères pour construire une représentation continue des entrées. Dans la deuxième partie de cette thèse, nous nous concentrons sur le développement de deux encodeurs en ligne : *l'encodeur min-hash*, basé sur une famille de fonctions de hash qui approxime localement la similarité entre deux chaînes de caractères ; et la *factorisation Gamma-Poisson*, un encodeur basé sur une factorisation matricielle qui considère une probabilité de distribution de la décomposition n-gram des catégories.

A l'aide d'expériences menées sur des données réelles et simulées, nous démontrons que les méthodes proposées dans le cadre de cette thèse améliorent l'apprentissage supervisé et ce, entre autre, du fait de leur capacité à capturer correctement l'information morphologique des entrées. Même avec des données volumineuses et non-traitées au préalable, ces méthodes s'avèrent être performantes, et dans le cas de la factorisation Gamma-Poisson, elles génèrent des vecteurs facilement interprétables. Par conséquent, nos méthodes peuvent être appliquées à l'apprentissage statistique automatique (AutoML) sans aucune intervention humaine.

CONTENTS

ACRONYMS

ANN	Approximate nearest neighbor
AutoML	Automated machine learning
KL	Kullback-Leibler
LDA	Latent Dirichlet allocation
LSA	Latent semantic analysis
LSH	Locality-sensitive hashing
NLP	Natural Language Processing
NMF	Non-negative matrix factorization
NMI	Normalized mutual information
PCA	Principal component analysis
RKHS	Reproducing kernel Hilbert space
SVD	Singular value decomposition
SVM	Support vector machine
Tf-idf	Term-frequency inverse-document-frequency
t-SNE	t-distributed stochastic neighbor embedding
VDM	Value difference metric

NOTATION

We write sets of elements with capital curly fonts, as $\mathcal{X}$. Elements of a vector space (we consider row vectors) are written in bold $\mathbf{x}$ with the i-th entry denoted by x_i, and matrices are in capital and bold $\mathbf{X}$, with x_{ij} the entry on the i-th row and j-th column.

Let C be a categorical variable such that $\text{dom}(C) \subseteq \mathcal{S}$, with $\mathcal{S}$ the set of finite length strings. We call *categorical entries* the elements of $\text{dom}(C)$. Let $s_i \in \mathcal{S}, i{=}1...n$, be the entry corresponding to the i-th sample of a dataset. For statistical learning, we want to find an encoding function $\text{enc}{:}\,\mathcal{S} \to \mathbb{R}^d$, such as $\text{enc}(s_i){=}\mathbf{x}_i$. We call $\mathbf{x}_i$ the *feature map* of s_i. To represent a string entry s, we introduce a count vector of its n-grams of consecutive characters $\mathbf{f} \in \mathbb{N}^m$, with m the number of different n-grams generated by the set of categorical entries in the training set. Table 0.1 contains a summary of the main variables used in this thesis.

Table 0.1 – Summary of notations

Symbol	**Name**	**Definition**
$\|\mathbf{x}\|_2$	Vector euclidean norm (ℓ_2 norm)	$\left(\sum_{i=1}^n x_i^2\right)^{1/2}$
$\|\mathbf{x}\|_1$	Vector/matrix ℓ_1 norm	$\sum_{i=1}^n \|x_i\|$
$\langle \mathbf{x}, \mathbf{y} \rangle$	Vector scalar product	$\sum_{i=1}^n x_i y_i$
$\|\mathbf{X}\|_F$	Matrix Frobenius norm	$\left(\sum_{i,j1=}^{n,m} x_{i,j}^2\right)^{1/2}$
$\text{Diag}(\mathbf{x})$	Diagonal matrix with diagonal $\mathbf{x}$	
$H(\mathbf{q})$	Shannon Entropy	$\sum_i^n q_i \log q_i$
$\triangle^k$	$(k-1)$-probability simplex	$\{\boldsymbol{\lambda} \in \mathbb{R}_+^k : \|\boldsymbol{\lambda}\|_1 = 1\}$
$\mathcal{N}(\mu, \boldsymbol{\Sigma})$	Normal distribution	
$\mathbb{P}[A]$	Probability of event A	
$\mathbb{E}[X]$	Expected value of X	$\sum_A \mathbb{P}[X = x]x$
$\mathcal{S}$	Set of all finite-length strings.	
$\mathcal{G}_n(s) \subseteq \mathcal{S}$	Set of all consecutive n-grams in $s \in \mathcal{S}$.	
$\mathcal{V}{=}\bigcup_{i=1}^n \mathcal{G}_n(s_i)$	Vocabulary of n-grams in the train set.	
C	Categorical variable.	
n	Number of samples.	
d	Dimension of the categorical encoder.	
$m = \|\mathcal{V}\|$	Cardinality of the vocabulary.	
$\mathbf{F} \in \mathbb{R}^{n \times m}$	Count matrix of n-grams.	
$\mathbf{X} \in \mathbb{R}^{n \times d}$	Feature matrix of C.	

1

OVERVIEW

Tabular datasets often contain columns with string entries. However, fitting statistical models on such data generally requires a numerical representation of all entries, which calls for building an *encoding*, or vector representation. Considering string entries as nominal—unordered—categories gives well-framed statistical analysis on this kind of data. In such situations, categories are assumed to be mutually exclusive and unrelated, with a small set of possible values.

Yet, in many real-world datasets, string columns are not standardized in a small number of categories. This poses challenges for statistical analysis. First, the set of all possible entries may be unknown and extremely large, as the number of different strings in the column can indefinitely increase with the number of data samples. We refer to this kind of columns as ***high-cardinality categorical variables***. Second, this high-cardinality usually generates categorical entries that may be related: they often carry some morphological or semantic overlap.

The classic approach to encode categorical variables for statistical analysis is *one-hot encoding* (Hastie et al., 2005). It creates vectors that agree with the general intuition about nominal categories: orthogonal and equidistant (P. Cohen et al., 2014). However, in high-cardinality cases, one-hot encoding leads to feature vectors of high dimensionality. This is particularly problematic in big data settings, which can lead to a very large number of categories, posing computational and statistical learning limitations.

Data engineering practitioners typically tackle these issues with data-cleaning techniques (Pyle, 1999; Rahm and Do, 2000): *deduplication* tries to merge different variants of the same entity (Christen, 2012b; Elmagarmid et al., 2007; Winkler, 2006), and *normalization*, used in databases and text processing, put string entries in canonical forms to reduce variability. A related concept is that of *feature engineering* (Domingos, 2012), where the idea is to manually build easy-to-learn features from the original data. However, all the previous techniques often require domain-specific human intervention, and are one of the most challenging and time-consuming tasks for data scientists [1].

In the context of statistical learning, given a prediction problem, variables can be either a *feature* variable (also known as explanatory, input or independent variable), or a *target* variable (also known as the response, output or dependent variable). In this thesis, we focus on the general problem of finding numerical representations for high-cardinality categorical variables that are part of the feature set. We

1. Kaggle industry survey: https://www.kaggle.com/surveys/2017

study new encoding approaches for statistical analysis on string categorical entries that are suited to a very large number of categories without any human intervention. Our goal is to provide feature vectors for categorical data that: *(i)* are of limited dimensionality; *(ii)* scale to large data settings; *(iii)* improve performance of statistical analysis tasks such as supervised learning; and *(iv)* do so without any cleaning, feature engineering, or neural architecture search. Also, as many machine-learning applications require algorithms that are understood or explainable by humans (Doshi-Velez and B. Kim, 2017), we focus on the interpretability of encoders.

To evaluate these new encoding methods, we collected 17 real-life datasets containing at least one high-cardinality categorical variable and one related prediction task. To our knowledge, this is the first collection of datasets related to this topic, as most machine learning repositories contain standard, low-cardinality, categorical variables [2]. All datasets are open source and we hope that they will foster more work on this subject.

This thesis is organized as follows. Chapter 2 introduces the supervised learning problem and formally defines categorical variables, the encoding process, as well as the problem of high-cardinality. Also, it presents and discusses the previous work in the subject. Chapter 3 (Cerda et al., 2018) proposes a generalization of one-hot encoding, *similarity encoding*, that considers pair-wise string similarities to build the encoding representations. Then, Chapter 4 explains similarity encoding as a *Fisher kernel*, a kernel method based on a generative process for character n-grams.

Classic encoding methods are hard to apply in online machine-learning settings. Indeed, new samples with new categories require recomputation of the encoding representation, and hence retrain the model from scratch. In the second part of this work we focus on online encoders that are scalable to large data settings (Cerda and Varoquaux, 2019). Chapter 5 presents the *min-hash encoder*, an stateless encoding method based on a family of hash functions that locally approximate similarity between strings (Broder, 1997). Then, Chapter 6 presents the *Gamma-Poisson factorization* (Canny, 2004), an encoder based on matrix factorization that considers a probability distribution of the n-gram decomposition of categories. An online algorithm for the Gamma-Poisson decomposition is developed for scalability concerns.

In the last part, we present the results of several benchmark experiments based on the 17 collected datasets, with focus on the prediction performance of encoders (Chapter 7) and interpretability (Chapter 8). Finally Chapter 9 gives the main conclusions of the thesis.

2. See for instance, the UCI Machine Learning Repository (Dheeru and Karra Taniskidou, 2017).

Part 1

GENERALIZING ONE-HOT ENCODING

2 BACKGROUND

2.1 CATEGORICAL VARIABLES

In classical statistical analysis, a ***categorical variable*** is typically defined as a non-numerical variable with values of either *ordinal* or *nominal* nature (Agresti and Kateri, 2011).

In the case of ordinal categorical variables, a total order between the values can be defined. For example, answers in the Likert scale (Likert, 1932) to the question: "*Do you agree with this statement: A child's education is the responsability of parents, not the school system*", compose an ordinal categorical variable in which the level of *agreement* can be associated with a numerical value. This intrinsic order makes ordinal categorical variables easy to analyze [1].

Conversely, *place of birth* is a nominal categorical variable, because there is no a clear order between cities that can be assigned a priori. In this thesis, we mainly focus on nominal categorical variables, because their numerical representation is much more challenging than in the ordinal case.

2.1.1 *Curated and non-curated categorical entries*

We say that a categorical variable has been *curated*, or *standardized* [2], when the set of different possible values is limited, known a priori —before the data collection process—, and with values that are assumed to be mutually exclusive. In this case, we call ***categories*** the different unique elements of the categorical variable.

For example, the variable *marital status* is a nominal categorical variable that can be standardized to 4 mutually exclusive categories: `married`, `single`, `divorced`, and `widowed`. After standardization, no other answers are allowed.

On the contrary, when a given categorical column has not been previously standardized, the set of possible responses is open—unknown a priori—, it typically grows with the number of samples, and the different values can contain semantical or morphological representations that are not mutually exclusive—there is overlap between the different values. In this case, we say that we are in presence of a *non-curated categorical variable*, and we call ***categorical entries***—or simply *entries*, for the purpose of this thesis—the different observed values.

Different entries can refer to the same category. For example, in a categorical column named *city*, the entries `"New York"`, `"New York city"`, and `"New York, NY"`, point to the same category, the "city of New York".

1. See Section 2.4.2 for the *ordinal encoding* method.
2. In the sense of making something conform to a standard.

Before extending the description of non-curated categorical variables in Section 2.5, the main object of interest this thesis, let us introduce in the next sections the basis of the of the statistical learning theory (see for example Hastie et al., 2005) and its relation with categorical variables in a table.

2.2 THE SUPERVISED LEARNING MODEL

Here we introduce some useful notions of statistical learning (Hastie et al., 2005; V. Vapnik, 2013). We focus on the *supervised learning model*, where the goal is to estimate the functional dependency between input values (or features) and output values (target) from a series of examples.

The *supervised learning model* is composed by three main parts:

(i) a *data generation process* of *i.i.d.* random vectors $\mathbf{x} \in \mathbb{R}^p$.

(ii) a *supervisor* that returns output values $y \in \mathcal{Y}$ following an unknown conditional probability distribution $\mathbb{P}(y|\mathbf{x})$.

(iii) a *learner*, capable to implement a parametrized set of functions $\{f_\alpha(\mathbf{x}) : \alpha \in \Lambda\}$[3], where $\boldsymbol{\alpha}$ are the parameters.

The *learning problem* is that of choosing—from the previous set of functions defined by the learner—the function $f_\alpha(\mathbf{x})$ which predicts the supervisors' response in the best possible way (V. N. Vapnik, 1999). In order to do this, the model considers n observations of the input and output random variables:

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n) \tag{2.1}$$

We call these observations the *training set*. Note that the input vector we consider is defined in the real space.

The type of *learning task* to solve depends on the nature of the output. When the output is numerical, we say that learning task is a *regression* problem. If the output is categorical, it is a *classification* problem, and we call *classes* the different values of the output.

Defining a a measure of discrepancy, or *loss*[4], $L(y, f_\alpha(\mathbf{x}))$, allows us to choose the best function by minimizing a *risk functional*:

$$R(\boldsymbol{\alpha}) = \int L(y, f_\alpha(\mathbf{x})) d\mathbb{P}(\mathbf{x}, y) \tag{2.2}$$

By using the training set, the risk functional can be approximated by the *empirical risk*:

$$\hat{R}(\boldsymbol{\alpha}) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f_\alpha(\mathbf{x}_i)) \tag{2.3}$$

To assess the *generalization* capacity of a model, an independent *testing set* of observations is usually drawn:

$$(\mathbf{x}'_1, y'_1), (\mathbf{x}'_2, y'_2), \ldots, (\mathbf{x}'_{n'}, y'_{n'}) \tag{2.4}$$

3. In practice, different function sets correspond to different learning algorithms.

4. The loss depends on the chosen learning algorithm and in the nature of the prediction problem: *regression*, *binary classification*, and *multiclass classification*.

The testing set is used to estimate the generalization of the model by using a *prediction metric*. Depending on the application and on the nature of the learning task, different prediction metrics can be considered [5].

In the next section we link the statistical learning framework to a database formalism (Cerda et al., 2018), which relies on sets. This is necessary in order to transform categorical variables into numerical inputs, as specified by the data generation process of the learning model.

2.3 FROM DATABASES TO STATISTICAL LEARNING

A table containing numerical and categorical columns is specified by its *relational scheme* $\mathcal{R}$:

$$\mathcal{R} = \{A_j, j = 1 \dots m\}, \tag{2.5}$$

the set of the A_j different attributes, *i.e.*, the column names of the table (Maier, 1983). Each attribute has a domain of possible values denoted by $\text{dom}(A_j) = \mathcal{D}_j$. Then, a table is defined as a *relation* r on the scheme $\mathcal{R}$:

$$r(\mathcal{R}) = \{t^i : \mathcal{R} \to \bigcup_{j=1}^{m} \mathcal{D}_j,\ i = 1 \dots n\}, \tag{2.6}$$

a set of *mappings* or (*m-tuples*), where for each *record* (or sample) $t^i \in r(\mathcal{R})$, $t^i(A_j) \in \mathcal{D}_j$, $j{=}1 \dots m$. If A_j is a numerical attribute, then $\text{dom}(A_j){=}\mathcal{D}_j \subseteq \mathbb{R}$. If A_j is a categorical attribute represented by strings, then $\mathcal{D}_j \subseteq \mathcal{S}$, where $\mathcal{S}$ is the set of finite-length strings [6]. As a shorthand, we call $k_j = \text{card}(\mathcal{D}_j)$, the cardinality of the categorical attribute A_j in the training set. Note that the domain can be extended if new unseen entries appear in the testing test.

As categorical entries are not numerical, they require an operation to define a feature matrix $\mathbf{X}$ from the relation r. Statistical or machine learning models that need vector data as input are applied after a ***categorical encoding***, a feature map that consists of replacing the samples $t^i(A_j), i = 1 \dots n$, by feature vectors:

$$\mathbf{x}_j^i \in \mathbb{R}^{d_j}, d_j \geqslant 1, \tag{2.7}$$

where d_j is the *dimensionality of the encoding* for the variable A_j. Using the same notation in case of numerical attributes, we can define $\mathbf{x}_j^i = t^i(A_j) \in \mathbb{R}^{d_j}$, with $d_j = 1$, and write the feature matrix $\mathbf{X}$ as:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^1 & \dots & \mathbf{x}_m^1 \\ \vdots & \ddots & \vdots \\ \mathbf{x}_1^n & \dots & \mathbf{x}_m^n \end{bmatrix} \in \mathbb{R}^{n \times p}, p = \sum_{j=1}^{m} d_j \tag{2.8}$$

5. Table 7.2 shows the metrics used in the empirical study of this tesis.

6. Some categorical variables can be represented by numbers, but if they are not considered to belong to set of real values, they are taken as strings. For instance, in the categorical variable *postal code*, values are usually represented by 5-digits numbers. However, a value greater than other is not necessarily informative, so they are considered as strings.

It is important to remark that in all the methods we study in this thesis, categorical variables are encoded independently to the others, as shown in Equation 2.8.

In standard supervised learning settings, the observations, represented by the feature matrix $\mathbf{X}$, are associated with a target vector $\mathbf{y} \in \mathbb{R}^n$ to predict. Usually, $\mathbf{y}$ corresponds to one of the columns of the original table.

2.4 ENCODING STRATEGIES FOR CURATED CATEGORICAL DATA

We now review classical encoding methods for categorical variables. These methods will be used as baseline in our empirical study (see Chapter 7). For simplicity of exposition, in the rest of this thesis we will consider only a single categorical variable $C \in \{A_j\}_{j=1}^n$, omitting the column index j from the previous definitions.

2.4.1 *One-hot encoding*

One-hot encoding is a simple and widely-used encoding method for standard categorical variables (Alkharusi, 2012; Berry et al., 1998; P. Cohen et al., 2014; Davis, 2010; Myers et al., 2010; O'Grady and Medoff, 1988; Pedhazur, Kerlinger, et al., 1973). For example, a categorical variable having as categories {`female`, `male`, `other`} can be encoded respectively with 3-dimensional feature vectors: {$[1,0,0]$, $[0,1,0]$, $[0,0,1]$}. In the resulting vector space, each category is orthogonal and equidistant to the others, which agrees with classical intuitions about nominal categorical variables (Section 2.1).

Let C be a categorical variable with cardinality $k \geqslant 2$ such that $\text{dom}(C)=\{s^{(\ell)}, 1 < \ell \leqslant k\}$ and $t^i(C)=s_i \in \text{dom}(C)$. The one-hot encoding method sets each feature vector as:

$$\mathbf{x}^i \stackrel{\text{def}}{=} \left[\mathbb{1}[s_i=s^{(1)}],\ \mathbb{1}[s_i=s^{(2)}],\ \ldots,\ \mathbb{1}[s_i=s^{(k)}]\right] \in \mathbb{R}^k \qquad (2.9)$$

where $\mathbb{1}[\cdot]$ is the indicator function. Several variants of one-hot encoding have been proposed[7], but in a linear regression, all perform equally in terms of the R^2 score[8] (see P. Cohen et al., 2014 for details).

LIMITATIONS. The one-hot encoding method is intended to be used when categories are mutually exclusive (P. Cohen et al., 2014), which is not necessarily true of non-curated data, as we previously saw in Section 2.1.1.

Another drawback of this method is that it provides no heuristics to assign an encoding vector to new categories that appear in the testing set but have not been encoded on the training set. Given the previous definition of Equation 2.9, the zero vector will be assigned

7. Variants of one-hot encoding include dummy coding, choosing the zero vector for a *reference* category, effects coding, contrast coding, and nonsense coding (P. Cohen et al., 2014).

8. The difference between methods is the interpretability, in the context of a linear regression, of the values for each variable.

to any new category in the testing set, which creates collisions if more that one new category is introduced. Additionally, high-cardinality categorical variables greatly increase the dimensionality of the feature matrix, which augments the computational cost of the learning algorithm. As we will see in Chapter 7, dimensionality reduction on the one-hot encoding vector tackles this problem, but with the risk of loosing information.

2.4.2 *Ordinal encoding*

In the case of ordinal categorical variables, a total order can be clearly identified between categories. Then, an effective encoding strategy consists in mapping this order to any subset of elements in $\mathbb{R}$. To do so, the simplest way is to transform ordinal categorical values into consecutive integer values. This method is called *ordinal encoding*. For example, the values of the Likert scale: *Strongly disagree*, *Disagree*, *Neither agree nor disagree*, *Agree* and *Strongly agree* are mapped respectively to the values: 0, 1, 2, 3, and 4, which preserves the intrinsic order in the degree of agreement. In general, ordinal encoding is not suitable for nominal categorical variables[9]. However, nominal variables can benefit of an order that is based on the effect of each category on the target, as proposed in the next encoding method.

2.4.3 *Encoding using target statistics*

Here we present two methods that use the statistics of the target variable. We start by the Value difference metric (VDM) (Duch et al., 2000), a strategy in which each category is encoded given the effect it has on the target variable $\mathbf{y}$. This creates an order in the real space that accounts for the expected value of the target, conditioned on each category $s^{(\ell)}$:

$$\mathbf{x}^i = \mathbb{E}_\ell\left[\mathbf{y}^\ell \,|\, s^{(\ell)}{=}s_i\right] \in \mathbb{R}. \tag{2.10}$$

Note that for regression and binary classification (with $y_i \in \{0, 1\}$), the obtained feature vector is one-dimensional. For the case of multiclass classification, the encoder has a dimension equal to the number of different classes.

A related encoding approach is the MDV continuousification scheme (Grabczewski and Jankowski, 2003), which encodes entries s_i by its expected value on each target c_k,

$$\mathbf{x}_i = \mathbb{E}_\ell\left[s^{(\ell)}{=}s_i \,|\, \mathbf{y}^\ell{=}c_k\right] \in \mathbb{R}. \tag{2.11}$$

In the case of a classification problem, c_k belongs to the set of possible classes for the target variable. However, in a dirty dataset, as with spelling mistakes, some categories can appear only once, undermining the meaning of their marginal link to the target.

9. but it can still be useful if used with certain type of non-linear algorithms.

2.5 ENCODING NON-CURATED CATEGORICAL VARIABLES.

With non-standardized categorical variables, the set of possible categories is unknown before the data collection process. One example of such non-standardized categories can be found in the Open Payments dataset [10], which describes financial relationships between healthcare companies and physicians or teaching hospitals. One possible task is to predict the value of the binary variable *status* (whether the payment has been done under a research protocol or not) given the following variables: *corporation name*, *(payment) amount*, and *dispute* (whether the physician refused the payment in a second instance). A challenge with this dataset is that some categories are not standardized. For instance, Table 2.1 shows all categories of the variable *company name* with the word `Pfizer` in it for the year 2013.

This type of data poses a problem from the point of view of the statistical analysis because we do not know a priori, without external expert information, which of these categories refer to the exact same company or whether all of them have slight differences and hence should be considered as different entities. Also, we can observe that the frequency of the different categories varies by several orders of magnitude, which could imply that errors in the data collection process have been made.

Table 2.1 – **Non-standardized categorical variables usually have entries that have common information.** Entities containing the word `Pfizer` in the variable *Company Name* of the Open Payments dataset (see Section 7.1 in the Appendix) and the respective number of times they appear in the dataset (year 2013).

Company name	Count
Pfizer Inc.	79,073
Pfizer Pharmaceuticals LLC	486
Pfizer International LLC	425
Pfizer Limited	13
Pfizer Corporation Hong Kong Limited	4
Pfizer Pharmaceuticals Korea Limited	3

Other examples of non-curated categorical variables are shown in Table 2.2. In the *Drug Directory* dataset [11], one of the variables is a categorical column with *non proprietary names* of drugs. As entries in this column have not been normalized, many of them are highly related: they share a common ingredient such as `alcohol` (see Table 2.2.a). Similarly, in the *Employee Salaries* dataset [12], where relevant variable is the *position title* of employees, there is also overlap in the different entries (Table 2.2.b).

10. https://openpaymentsdata.cms.gov/.

11. Product listing data for all unfinished, unapproved drugs. Source: U.S. Food and Drug Administration (FDA)

12. Annual salary information for employees of the Montgomery County, MD, U.S.A. Source: https://data.montgomerycountymd.gov/

Table 2.2 – **Examples of high-cardinality categorical variables.**

Non Proprietary Name	Count
alcohol	1736
ethyl **alcohol**	1089
isopropyl **alcohol**	556
polyvinyl **alcohol**	16
isopropyl **alcohol** swab	12
62% ethyl **alcohol**	12
alcohol 68%	6
alcohol denat	6
dehydrated **alcohol**	5

(a) Count for some of the categories containing the word `alcohol` in the *Drug Directory* dataset. The dataset contains more than 120k samples.

Employee Position Title
Police Aide
Master **Police** Officer
Mechanic **Technician** II
Police Officer III
Senior Architect
Senior Engineer **Technician**
Social Worker III
Bus Operator

(b) Some categories in the *Employee Salaries* dataset. For 10 000 employees, there are almost 400 different occupations. Yet, they share relevant substrings.

2.5.1 *Sources of high cardinality*

Often, the cardinality of a dirty categorical variable grows with the number of samples in the dataset. Figure 2.1 shows the cardinality of the corresponding categorical variable as a function of the number of samples for each of the 17 datasets that we analyze in this paper.

Dirty categorical data can arise from a variety of mechanisms (W. Kim et al., 2003; Oliveira et al., 2005):

- Typographical errors (*e.g.*, `proffesor` instead of *professor*)
- Extraneous data (*e.g.*, name and title, instead of just the name)
- Abbreviations (*e.g.*, `Dr.` for `doctor`)
- Homonyms (*e.g.*, `Montréal`, of Canada or France)
- Aliases (*e.g.*, `Ringo Starr` instead `Richard Starkey`)
- Encoding formats (*e.g.*, ASCII, EBCDIC, etc.)
- Uses of special characters (space, colon, dash, parenthesis, etc.)
- Concatenated hierarchical data (e.g., state-county-city vs. state-city)

The presence of a large number of categories calls for representing the relationships between them. In knowledge engineering this is done via an ontology or a taxonomy. When the taxonomy is unknown, the problem is challenging. For example, in the *Medical Charges* dataset, `cervical spinal fusion` and `spinal fusion except cervical` are different entries, but both share the fact that they are a spinal fusion, hence they are closely related.

We now review encoding strategies in the presence of dirty or high-cardinality categorical data.

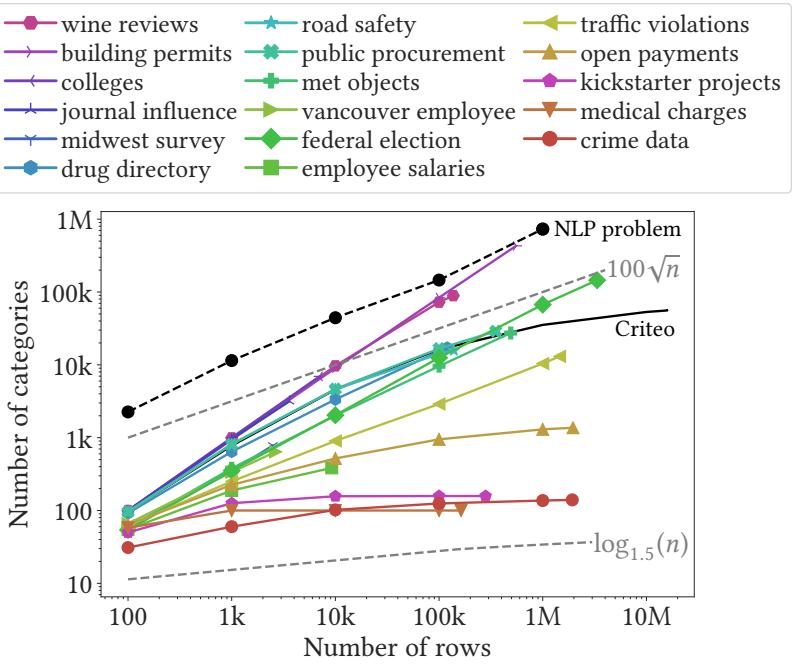


Figure 2.1 – **Number of categories versus number of samples**. In most datasets, a higher number of samples implies a higher number of categories for the respective variable. In general, the cardinality of categories grows slower than words in a typical NLP problem (in this case, the first paragraph of random Wikipedia articles). A dataset description can be found in Section A.1 in the Appendix.

2.5.2 *Strategies for entries without subword information*

In some cases, the subword information of categorical entries it is not relevant. For instance, in categorical variables as *Country name*, the overlap of character n-grams does not have a relevant meaning. This section describe some encoding methodologies for this kind of data.

2.5.2.1 *Target encoding*

The target encoding method (Micci-Barreca, 2001), is a variation of the VDM *continuousification scheme* (Duch et al., 2000), in which each category is encoded given the effect it has on the target variable $\mathbf{y}$ [13]. The method considers that categorical variables can contain rare categories. Hence it represents each category by the probability of $\mathbf{y}$ conditional on this category. In addition, it takes an empirical Bayes approach to shrink the estimate. Thus, for a binary classification task:

$$\mathbf{x}^i = \lambda(f^i)\,\mathbb{E}_\ell\left[\mathbf{y}^\ell \,\middle|\, s^{(\ell)}{=}s_i\right] + \left(1-\lambda(f^i)\right)\mathbb{E}_\ell\left[\mathbf{y}^\ell\right] \quad \in \mathbb{R} \tag{2.12}$$

13. Several variations of this idea are presented in Prokhorenkova et al., 2018. However, none of them account for the morphology of string categories, and we do not explore them in this thesis.

where f^i is the frequency of the category $s^{(i)}$ and $\lambda(f^i) \in [0, 1]$ is a weight such that its derivative with respect to n^i is positive, e.g.:

$$\lambda(f^i) = \frac{1}{1 + e^{-\frac{f^i - a}{b}}}, \tag{2.13}$$

where a and b are parameters to tune. Another option with only one parameter is:

$$\lambda(f^i) = \frac{f^i}{f^i + a}, \; a > 0. \tag{2.14}$$

Note that the obtained feature vector is one-dimensional for the case of regression and binary classification [14].

As an example, Figure 2.2 shows the encoding values for different entries in the *Employee Salaries* dataset for the categorical variable *Occupation*. It can be observed that manager positions are encoding with higher values.

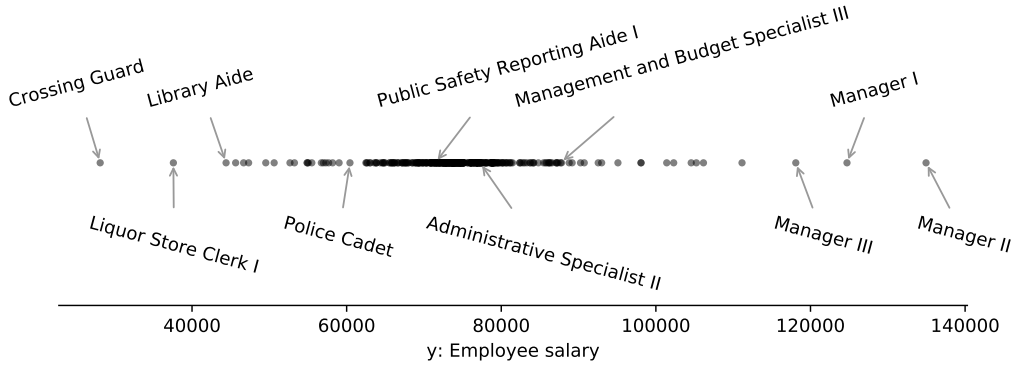


Figure 2.2 – **The target encoding maps categories into single dimensions according to its expected target value.** Expected annual salary for categories in the Employee Salaries dataset.

The biggest limitation of the target encoding for our particular setting, is that is does not consider at all the subword information of categorical entries.

2.5.2.2 *Hash encoding*

A solution to reduce the dimensionality of the data is to use the hashing trick (Weinberger et al., 2009). Instead of assigning a different unit vector to each category, as one-hot encoding does, one could define a hash function to designate a feature vector on a reduced vector space. This method does not consider the problem of dirty data either, because it assigns hash values that are independent of the morphological similarity between categories.

14. In the case of multi-class classification with ℓ labels, the same idea can be extended to ℓ "*one v/s all*" binary classification problems, and the final encoding will have a dimensionality equal to ℓ.

2.5.3 *Strategies for entries with meaningful subword structure*

In many cases categorical entries present useful information in their string representation, as in the datasets collected for this study. In these cases, it may be appropiate to consider encoding methods based on the substring components of entries. Here, we present some of these approaches.

2.5.3.1 *Bag-of-words, bag-of-n-grams, and tf-idf*

A simple way to capture morphology in text documents is to characterize it by the count of its words. This method is usually called *bag-of-words*. It assumes that, in order to characterize a given document, the order of words in it is not necessarily relevant. In the language of probability theory, this is an assumption of exchangeability for the words in a document (Aldous, 1985).

For high-cardinality categorical variables, the number of different n-grams tends to increase with the number of samples. Yet, this number increases slower than in a typical Natural Language Processing (NLP) problem (see Figure 2.3). Indeed, categorical variables have less entropy than free text: they are usually repeated, often have subparts in common, and refer to a particular, more restrictive subject.

A simple way to capture morphology in a string is to characterize it by the count of its character n-grams. This is sometimes called a *bag-of-n-grams* characterization of strings. Such representation has been shown to be efficient for spelling correction (Angell et al., 1983) or for named-entity recognition (Klein et al., 2003).

Representing strings by character-level n-grams is related to vectorizing text by their tokens or words. Common practice uses Term-frequency inverse-document-frequency (Tf-idf) reweighting: dividing a token's count in a sample by its count in the whole document (Salton and McGill, 1983). Dimensionality reduction by a Singular value decomposition (SVD) on this matrix leads to a simple topic extraction, Latent semantic analysis (LSA) (Deerwester et al., 1990; Landauer et al., 1998). A related but more scalable solution for dimensionality reduction are random projections, which give low-dimensional approximation of Euclidean distances (Achlioptas, 2003; Johnson and Lindenstrauss, 1984).

2.5.3.2 *Neural networks for encoding categories*

Guo and Berkhahn, 2016, proposes an encoding method based on neural networks. It is inspired by NLP methods that perform word embedding based on textual context (T. Mikolov et al., 2013). In tabular data, the equivalent to this context is given by the values of the other columns, categorical or not. The approach is simply a standard neural network, trained to link the table $\mathcal{R}$ to the target $\mathbf{y}$ with standard supervised-learning architectures and loss and as inputs the table with categorical columns one-hot encoded. Yet, Guo and Berkhahn, 2016 uses as a first hidden layer a bottleneck for each categorical variable. The corresponding intermediate representation,

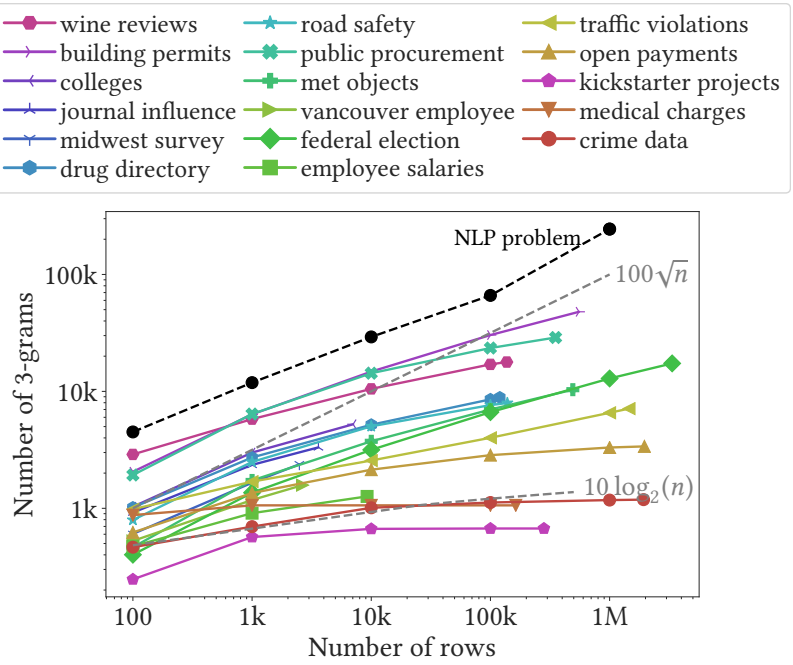


Figure 2.3 – **Number of 3-gram versus number of samples**. The number of different n-grams tends to increase slower than in a typical NLP problem (in this case, the first paragraph of random Wikipedia articles).

learned by the network, gives a vector embedding of the categories in a reduced dimensionality. This approach is interesting as it guides the encoding in a supervised way. Yet, it entails the computational and architecture-selection costs of deep learning. Additionally, it is still based on an initial one-hot encoding which is susceptible to dirty categories.

2.5.3.3 *Word embeddings*

If the string entries are common words, an approach to represent them as vectors is to leverage word embeddings (T. Mikolov et al., 2013; Pennington et al., 2014). Euclidean similarity of these vectors captures related semantic meaning in words. Multiple words can be represented as a weighted sum of their vectors, or with more complex approaches (Arora et al., 2016). To cater for out-of-vocabulary strings, FastText (Bojanowski et al., 2017) considers subword information of words, *i.e.*, character-level n-grams. Hence, it can encode strings even in the presence of typos. Word vectors computed on very large corpora are available for download. These have captured fine semantic links between words. However, to analyze a given database, the danger of such approach is that the semantic of categories may differ from that in the pretrained model. These encodings do not adapt to the information specific in the data at hand. Moreover, they cannot be trained directly on the categorical variables for two reasons: categories lack of enough context, as they are usually composed of short strings; and the number of samples in some datasets is not enough to properly train these models.

2.6 FORMATTING AND CURATING STRING ENTRIES.

Finally, we present some data-cleaning techniques to tackle variability in string entries.

2.6.1 *Cleaning techniques in databases*

STEMMING OR LEMMATIZING. Stemming and lemmatizing are text preprocessing techniques that strive to extract a common root from different variants of a word (Hull et al., 1996; Lovins, 1968). For instance, 'standardization', 'standards', and 'standard' could all be reduced to 'standard'. These techniques are based on a set of rules, crafted to the specificities of a language. Their drawbacks are that they may not be suited to a specific domain, such as medical practice, and are costly to develop. Some recent developments in NLP avoid stemming by working directly at the character level (Bojanowski et al., 2017).

DEDUPLICATION / RECORD LINKAGE In databases, deduplication or record linkage strives to find different variants that denote the same entity and match them (Elmagarmid et al., 2007). Classic record linkage theory deals with merging multiple tables that have entities in common. It seeks a combination of similarities across columns and a threshold to match rows (Fellegi and Sunter, 1969). If known matching pairs of entities are available, this problem can be cast as a supervised or semi-supervised learning problem (Elmagarmid et al., 2007). If there are no known matching pairs, the simplest solution boils down to a clustering approach, often on a similarity graph, or a related expectation maximization approach (Winkler, 2002). Supervising the deduplication task is challenging and often calls for human intervention. Sarawagi and Bhamidipaty, 2002, uses active learning to minimize human effort. Much of the recent progress in database research strives for faster algorithms to tackle huge databases (Christen, 2012a).

3 SIMILARITY ENCODING

Here we present a generalization of the commonly used one-hot encoding method (see Section 2.4.1 for a formal definition), that we call *similarity encoding*. Similarity encoding has been shown to be specially suitable for string categorical variables with a high cardinality (Cerda et al., 2018). The word *similarity* comes from the notion of *string similarity* (Navarro, 2001) between pairs of strings, that it takes the value 1 if the strings are identical, and it goes towards 0 for very different strings. String similarities are commonly used in the entity resolution literature (W. W. Cohen et al., 2003). Some of the most important ones are defined later in this chapter.

3.1 INTUITIONS

The one-hot encoding representation for a given category can be seen as a feature vector in which each dimension corresponds to the zero-one similarity between the entry we want to encode and all the known categories: it takes the value 1 when it is equal to the corresponding category and 0 if not (see Equation 2.9). When some form of morphological overlap is encountered between the categorical entries, one would like to extend the notion of one-hot encoding to a less restrictive vector representation. To improve the intuitions behind the previous idea, let us continue with an example.

Consider a categorical variable called *City of Residence*, composed by the following 3 possible entries: `London`, `Londres`, and `Paris`. The respective one-hot encoding (row) feature vectors are represented in the following Table 3.1:

Table 3.1 – Feature vectors with **one-hot encoding**. Rows represent the samples (and its respective feature vectors) and columns denote the corresponding feature names.

	City of Residence		
	London	Londres	Paris
`Londres`	0	1	0
`London`	1	0	0
`Paris`	0	0	1

In this example, the entries `London` and `Londres` point to the same entity, the "city of London". Performing data-cleaning would probably modify the entry `Londres` to its English translation `London`, as shown on Table 3.2. The problem with this cleaning step is that some information is lost: "Londres" is the French (or Spanish) translation of London. In this particular example, the original answers give an

Table 3.2 – Feature vectors with **deduplication**. Entries were manually translated to English.

	City of Residence	
	London	Paris
`London`	1	0
`London`	1	0
`Paris`	0	1

insight about the foreign origin of the respondent, which could be a potentially useful information to be exploited in a related learning problem (for example, predicting the annual salary of respondents). Nevertheless, we would still like to express the close relation between these two entries. As proposed before, one-hot encoding can be extended to a more general version that considers continuous string similarities instead. With this kind of approach, the entries `Londres` and `London` should not be merged; a closer representation in the feature space-for example, in terms of the ℓ_2 norm—should be enough to improve the generalization of the corresponding learning problem. For instance, the one hot encoding example of Table 3.1 can be transformed into the following continuous representation:

Table 3.3 – Feature vectors with similarity encoding.

	City of Residence		
	London	Londres	Paris
`Londres`	0.8	1	0
`London`	1	0.8	0
`Paris`	0	0	1

In this case, no information is lost, but `Londres` and `London` are closer to each other, with an arbitrary similarity value equal to 0.8.

The following section defines and explores different string similarity measures.

3.2 STRING SIMILARITIES

Let $\text{sim}: \mathcal{S} \times \mathcal{S} \to [0, 1]$ be an arbitrary string similarity measure satisfying the following properties:

- $\forall s_1, s_2 \in \mathcal{S},\ s_1 = s_2 \Rightarrow \text{sim}(s_1, s_2) = 1,$ (3.1)
- $\forall s_1, s_2 \in \mathcal{S},\ \text{sim}(s_1, s_2) = \text{sim}(s_2, s_1).$ (3.2)

A survey of text similarity measures can be found in W. W. Cohen et al., 2003; Gomaa and Fahmy, 2013. Most of these similarities are based on a morphological comparison between two strings. We describe in the next section some of the most commonly used similarity measures.

3.2.1 *Levenshtein*

It is based on the Levenshtein distance (Levenshtein, 1966) dist_{lev} between two strings $s_1, s_2 \in \mathcal{S}$, and it belongs to a more general family of distances, called *edit distances* (Navarro, 2001). It is calculated as a function of the minimum number of edit operations (*deletion*, *insertion* and *replacement*) that are necessary to transform one string into the other [1]. The Levenshtein distance has been widely applied in different domains, from record linkage (Conrad et al., 2016), spelling correction (Brill and Moore, 2000) and linguistics (Serva and Petroni, 2008).

In this work, we arbitrarily use a Levenshtein distance in which all edit operations have a weight of 1, except for the *replacement* operation, which has a weight of 2. A recursive definition of the Levenshtein distance is given by:

$$\text{dist}_{\text{lev}}(s_1, s_2) = \text{lev}_{s_1,s_2}(|s_1|, |s_2|), \tag{3.3}$$

with:

$$\text{lev}_{s_1,s_2}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j)=0, \\ \min \begin{cases} \text{lev}_{s_1,s_2}(i\text{-}1, j) + 1 \\ \text{lev}_{s_1,s_2}(i, j\text{-}1) + 1 \\ \text{lev}_{s_1,s_2}(i\text{-}1, j\text{-}1) + 2\,\mathbb{1}[s_{1i} \neq s_{2j}] \end{cases} & \text{otherwise.} \end{cases} \tag{3.4}$$

where $|s|$ is the character length of the string s. Then, we can obtain a similarity measure using the following transformation:

$$\text{sim}_{\text{lev}}(s_1, s_2) = 1 - \frac{\text{dist}_{\text{lev}}(s_1, s_2)}{|s_1| + |s_2|} \tag{3.5}$$

3.2.2 *Jaro-Winkler*

The Jaro-Winkler similarity (Winkler, 1999) is a variation of the Jaro distance $\text{dist}_{\text{jaro}}$ (Jaro, 1989), a metric that only considers the *transposition* operation, in contrast with the three edit operations allowed by the Levenshtein distance. The Jaro distance is defined as follows:

$$\text{dist}_{\text{jaro}}(s_1, s_2) = \frac{m}{3|s_1|} + \frac{m}{3|s_2|} + \frac{m-t}{3m}, \tag{3.6}$$

where m is the number of matching characters between s_1 and s_2 [2], and t is the number of character transpositions between the strings

1. There are several variations of the Levenshtein distance, that include different edit operations. The Damerau–Levenshtein distance (Damerau, 1964) allows also the operation *transposition* of adjacent characters; the longest common subsequence (LCS) distance does not allows *replacement*; and the Hamming distance (Hamming, 1950) only allows *replacement*. In this thesis we do not explore all these variations.

2. Two characters belonging to s_1 and s_2 are considered to be a match if they are identical and the difference in their respective positions does not exceed $2\max(|s_1|, |s_1|) - 1$. For m=0, the Jaro distance is set to 0.

s_1 and s_2 without considering the unmatched characters. Then, the Jaro similarity is defined as:

$$\text{sim}_{\text{jaro}} = 1 - \text{dist}_{\text{jaro}} \tag{3.7}$$

The Jaro-Winkler similarity $\text{sim}_{\text{j-w}}$ emphasizes prefix similarity between the two strings. It is defined as:

$$\begin{aligned} \text{sim}_{\text{j-w}}(s_1, s_2) &= \text{sim}_{\text{jaro}} + \ell\, p\, (1 - \text{sim}_{\text{jaro}}) && (3.8)\\ &= 1 - \text{dist}_{\text{jaro}}(s_1, s_2) + \ell\, p\, \text{dist}_{\text{jaro}}(s_1, s_2) && (3.9) \end{aligned}$$

where ℓ is the length of the longest common prefix between s_1 and s_2, and p is a constant scaling factor.

3.2.3 *N-gram similarity*

The n-gram similarity assumes that text can be characterized by the set of substrings that are contained on them. This similarity is based on splitting both strings into n-grams of consecutive characters and then calculating the Jaccard coefficient between them (Angell et al., 1983; Ukkonen, 1993)[3]:

$$\text{sim}_{\text{n-gram}}(s_1, s_2) = \frac{|\mathcal{G}_n(s_1) \cap \mathcal{G}_n(s_2)|}{|\mathcal{G}_n(s_1) \cup \mathcal{G}_n(s_2)|}, \tag{3.10}$$

where $\mathcal{G}_n(s), s \in \mathbb{S}$, is the set of all n-grams—an n-gram is a chain of consecutive characters of size n—contained in the string s. For instance, the 3-gram sets for the entries `Paris` and `Parisian` are:

$$\begin{aligned} \mathcal{G}_3(\texttt{Paris}) &= \{\texttt{Par}, \texttt{ari}, \texttt{ris}\},\\ \mathcal{G}_3(\texttt{Parisian}) &= \{\texttt{Par}, \texttt{ari}, \texttt{ris}, \texttt{isi}, \texttt{sia}, \texttt{ian}\}. \end{aligned}$$

These words have three 3-grams in common over a total of six different ones. Therefore, their similarity is:

$$\text{sim}_{\texttt{3-gram}}(\texttt{Paris}, \texttt{Parisian}) = \frac{1}{2}.$$

Figure 3.1 shows the distribution of similarity values, in logarithmic scale, for 10,000 random pairs of categorical entries in different datasets. In most of the datasets, the n-gram similarity has a median similarity value close to 0. Thus, in a similarity encoding setting, the n-gram similarity does not completely removes the sparseness obtained with one-hot encoding.

3. There exist more efficient versions of the 3-gram similarity (Kondrak, 2005), but we do not explore them in this work.

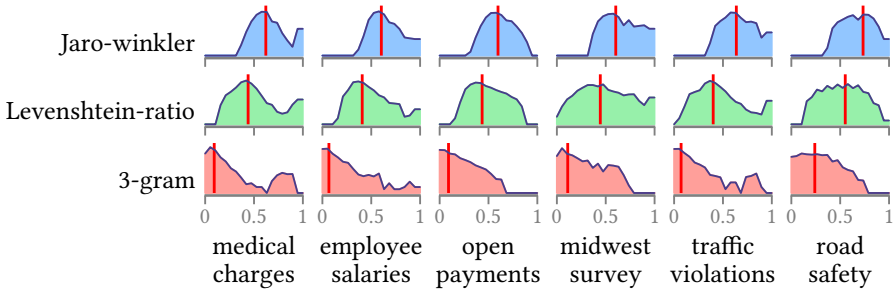


Figure 3.1 – **Histogram of pairwise similarity between categories for different string similarity metrics.** 10,000 pairs of categories were randomly generated for each dataset (y-axis in logarithmic scale). The red bar denotes the median value for each distribution. Note that *medical charge*, *employee salaries* and *traffic violations* present bimodal distributions.

3.3 SIMILARITY ENCODER

Given a similarity measure, one-hot encoding can be generalized to account for similarities in categorical entries. Let C be a categorical variable of cardinality k, and let sim be a similarity measure that satisfies the properties in Equations 3.1 and 3.2. The similarity encoding we propose replaces each of the instances of C, s_i, $i{=}1...n$ by a feature vector $\mathbf{x}^i \in \mathbb{R}^k$ so that:

$$\mathbf{x}_i \stackrel{\text{def}}{=} \left[\text{sim}(s_i, s^{(1)}),\ \text{sim}(s_i, s^{(2)}),\ \ldots,\ \text{sim}(s_i, s^{(k)})\right] \in \mathbb{R}^k. \quad (3.11)$$

Here, the elements $\{s^{(j)} \in \mathcal{S}, i = 1 \ldots k\}$ are a subset of the elements in $\text{dom}(C)$. By default, all categories in the training set can be taken, as with one-hot encoding.

With the previous definition, one-hot encoding corresponds to taking the discrete string similarity:

$$\text{sim}_{\text{one-hot}}(s_i, s_j) = \mathbb{1}[s_i = s_j] \quad (3.12)$$

where $\mathbb{1}[\cdot]$ is the indicator function.

Note that, from a kernel point of view (Schölkopf and Smola, 1998), Equation 3.11 defines the following kernel between two given observations:

$$\langle s_i, s_j \rangle_{\text{sim}} = \sum_{l=1}^{k} \text{sim}(s_i, s^{(l)})\, \text{sim}(s_j, s^{(l)}) \quad (3.13)$$

Hence, it projects on a dictionary of reference n-grams and gives more importance to the n-grams that best capture the similarity between categories. This differs from a simple string kernel that only takes into accounts pair-wise similarities, without considering the rest of the entries.

Finally, another advantage of this generalization over one-hot encoding is that new entries that are not present in the test set are naturally encoded without creating collisions. This avoids recomputation of the encoding representation, and hence retrain the learning model from scratch.

3.4 LIMITATIONS: HANDLING HIGH-DIMENSIONALITY

With one-hot or similarity encoding, high-cardinality categorical variables lead to high-dimensional feature vectors. This may lead to computational and statistical challenges: the encoding step may take a lot of time, but also the learning step. To tackle this issue, dimensionality reduction may be used on the resulting feature matrix. A natural approach is to use Principal component analysis (PCA), as it captures the maximum-variance subspace. Yet, it entails a high computational cost[4] and is cumbersome to run in an online setting. Hence, we explore three dimensionality reduction methods: *random projections*, the *most frequent* categories and a *k-means* based method.

RANDOM PROJECTIONS. Based on the Johnson-Lindenstrauss lemma, random projections give a reduced representation that accurately approximates distances of the original feature space (Rahimi and Recht, 2008). A drawback of such a projection approach is that it requires first computing the similarity to all categories. Also, it mixes the contribution of all categories in non trivial ways and hence may make interpreting the encodings difficult.

For this reason, we also explored methods based on a reduced set of prototypes: choosing a small number d of categories and encoding by computing the similarity to these prototypes only. These prototypes should be representative of the full category set in order to have a meaningful reduced space. This strategy has the advantage of reducing both the encoding and learning steps.

MOST FREQUENT CATEGORIES. One simple approach is to choose the $d \ll k$ most frequent categories of the dataset. This strategy makes sense because, in general, the most repeated entries are the ones that do not contain any typographical errors.

K-MEANS. Another way of choosing prototype elements in the category set are clustering methods like k-means, which chooses cluster centers that minimize a distortion measure. We use as prototype candidates the closest element to the center of each cluster. Note that we can apply the clustering on a initial version of the similarity-encoding matrix computed on a subset of the data. Clustering of dirty categories based on a string similarity is strongly related to deduplication or record-linkage strategies used in database cleaning. One notable difference with using a cleaning strategy before statistical learning is that we are not converting the various forms of the categories to the corresponding cluster centers, but rather encoding their similarities to these.

Finally, another drawback of similarity encoding—as defined in this chapter—is that it seems to be a purely heuristic method, without any theoretical support. We study this issue in the next chapter.

4. Precisely, the cost of PCA is $\mathcal{O}(n\,p\,\min(n,p))$.

4

FISHER KERNELS: EXPLAINING SIMILARITY ENCODING

Here we use a probabilistic model of categorical string entries to explain why the form of the representations used in similarity encoding is useful.

4.1 FISHER KERNELS

Kernels in machine learning are used to build a discriminant function by specifying only a similarity between instances (Schölkopf and Smola, 1998). They rely on the fact that, for certain learning problems, as with Support vector machines (SVM's), ridge, or logistic regression, the cost function and its solution can be entirely expressed in terms of the kernel evaluated on the data (Schölkopf and Smola, 1998).

Kernels have been widely used with objects where there is no obvious numerical representation, such as genomes or proteins in biology (Eskin et al., 2003), text (Gärtner, 2003; Lodhi et al., 2002) or images (H. Wang and Jingbin Wang, 2014). Using the kernel to train a model entails an n^2 cost in the number of samples of the training set. Therefore, with the current trend of learning in increasingly large data, learning with kernels has become less popular. However, a kernel defines implicitly a Reproducing kernel Hilbert space (RKHS) (Schölkopf and Smola, 1998), in which each sample is associated to a vectorial feature map. The kernel is then replaced by the inner product of the feature maps $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$, and learning can be performed on these vectorial data, for instance using stochastic optimization for very large data. Even when feature maps are used, instead of kernels, kernels provide a good conceptual framework to consider vectorizing non numerical data.

One way to build useful kernels is to consider a generative process of the data, that can be used to define a *Fisher kernel* (T. Jaakkola and Haussler, 1999). The central argument here is that a parametrized probabilistic model of the data defines a natural distance between instances based on how the parametrization affects their respective likelihoods. More precisely, let the model be specified by the probability $\mathbb{P}(\mathcal{X}_i|\theta)$ for a sample $\mathcal{X}_i$, where $\theta \in \mathbb{R}^p$ parametrizes the model. A natural distance between two data points $\mathcal{X}_1$ and $\mathcal{X}_2$ should capture the variations in their likelihood with respect to small variations in the parametrization. Defining a fully-fledged distance requires concepts of information geometry (Amari and Nagaoka, 2007): the natural distance is built by integrating a metric using the Fisher information matrix, *i.e.*, the expectation of Hessian of the log-likelihood. However, Fisher kernels focus on local properties, for which there is

no need to integrate. In their simplest form, they use as a feature map the Fisher score:

$$\mathbf{u}(\mathcal{X}_i) = \nabla_\theta \log p(\mathcal{X}_i|\theta), \quad \mathbf{u}(\mathcal{X}_i) \in \mathbb{R}^p \tag{4.1}$$

The inner product $\langle \mathbf{u}(\mathcal{X}_i), \mathbf{u}(\mathcal{X}_j) \rangle$ then locally captures the essence of the natural distance and is simpler to compute.

Typically, the parameters of the probabilistic model are estimated on the data at hand, *e.g.*, by using a training set, and then the gradient in Equation 4.1 is used to compute a vectorial representation $\mathbf{u}(\mathcal{X}_i)$ for each sample $\mathcal{X}_i$. This two-step procedure to build a vectorization is well known to text processing, *e.g.*, to build a tf-idf representation of documents: a first pass on the corpus defines the vocabulary and the frequency of the terms across the documents: then the vectorial representation can be computed. Indeed, Elkan, 2005, shows that modeling term occurrences as a Dirichlet compound multinomial distribution yields a Fisher kernel similar to the kernel of a tf-idf representation of documents. Fisher kernels can thus provide a deeper theoretical understanding of well known feature-extraction methods.

Other applications of Fisher kernels are visual vocabularies for image classification (Perronnin and Dance, 2007), audio classification (Moreno and Rifkin, 2000) and the discovery of protein homologies [1] in biology (T. S. Jaakkola et al., 1999).

4.2 SIMPLE STRING KERNEL OF INDEPENDENT N-GRAMS

Here we craft an encoding method from a Fisher kernel. For this, we model the n-gram vectors by extrapolating n-grams distributions of categories.

For a categorical variable C of known domain, a categorical distribution [2] gives the probability of its elements $s^{(i)} \in \text{dom}(C)$:

$$p\left(s^{(i)}|\theta = (\theta_1, \ldots, \theta_k)\right) = \prod_{l=1}^{k} \theta_l^{\mathbb{1}[s^{(i)}=s^{(l)}]}, \quad i = 1 \ldots k. \tag{4.2}$$

where θ_l is the frequency of the category $s^{(l)}$. Given a training set, this distribution is simply estimated by computing the frequencies of the observed categories. In practice, to reduce the number of parameters, the frequencies can be alternatively computed in a subset of prototype categories, chosen heuristically in the same way as for similarity encoding (Cerda et al., 2018).

The categorical distribution focuses all the mass on the prototype categories: it assigns a zero probability to an unseen string. Thus, it does not enable us to generalize across unseen categories. To work around this problem, we reformulate it as a distribution on the n-gram vector $\mathbf{f}$. We then relax the model to extrapolate to unseen

1. A protein homology is a shared sequence of amino acids (the building blocks of proteins) between different living organisms that can link them to a common ancestor.

2. A categorical distribution is the generalization the Bernouilli to multiple values.

n-gram vectors, using a form of smoothing, as in a kernel density estimator.

We now derive the distribution on the n-gram vectors, $p(\mathbf{f}|\theta)$. To account for possible morphological variations of $s^{(i)}$, we need to consider variations of individual elements of $\mathbf{f}$. The simplest assumption is that the dimensions of $\mathbf{f}$ are independent, with:

$$\begin{aligned} p(f_j, s^{(i)}|\theta) &= p(f_j|s^{(i)})\, p(s^{(i)}|\theta) \\ &= \theta_i\, p(f_j|s^{(i)}) \end{aligned} \tag{4.3}$$

Note that the probability of a given n-gram count f_j is completely determined if it is conditioned to being the representation of a prototype category. In other words, we can write:

$$p(f_j|s^{(i)}) = \mathbb{1}[f_j = f_j^{(i)}] \tag{4.4}$$

Integrating $p(f_j, s^{(i)}|\theta)$ in Equation 4.3 over the prototype categories gives:

$$p(f_j|\theta) = \sum_{i=1}^{k} \theta_i\, \mathbb{1}[f_j = f_j^{(i)}] \tag{4.5}$$

Finally, considering that the n-gram dimensions are independent, the joint probability of the count vector is:

$$\begin{aligned} p(\mathbf{f}|\theta) &= \prod_{j=1}^{m} p(f_j|\theta) \\ &= \prod_{j=1}^{m} \sum_{i=1}^{k} \theta_i\, \mathbb{1}[f_j = f_j^{(i)}] \end{aligned} \tag{4.6}$$

The latter expression is a *naive* generative model, as it considers that n-gram counts are independent and entirely defined by the number of times this count value is observed in the set of prototype categories. From a conceptual standpoint, it consists in extrapolating the observed distribution with a more entropic one.

The Fisher score of this distribution can now be used to derive a Fisher-kernel feature map, which gives our category encoding:

$$\begin{aligned} u_i(s) &= \frac{\partial}{\partial \theta_i} \log p(\mathbf{f}|\theta) \\ &= \sum_{j=1}^{m} \frac{\mathbb{1}[f_j = f_j^{(i)}]}{\sum_{l=1}^{k} \theta_l\, \mathbb{1}[f_j = f_j^{(l)}]}, \quad i = 1 \dots k. \end{aligned} \tag{4.7}$$

4.3 LINK TO SIMILARITY ENCODING

The encoding of Equation 4.7 can be evaluated on any string s. It is equal to 0 if s and $s^{(i)}$ have no n-grams—or absence of n-grams—in common, and is upper bounded by the Fisher score of the respective

prototype $u_i(s^{(i)})$. Thus, rescaling the Fisher scores defines a string similarity:

$$\text{sim}_{\text{naive}}(s, s^{(i)}) \stackrel{\text{def}}{=} \frac{u_i(s)}{u_i(s^{(i)})} \tag{4.8}$$

To better understand the expression of Equation 4.7, consider that all category frequencies θ_i are the same. The similarity then sums the number of times the string represented by $\mathbf{f}$ has the same number of a given n-gram f_j as the category i, normalized by the count of this specific value of f_j across all categories.

Note that this similarity takes into account the entire n-gram vocabulary, and not only the n-gram set generated by the two compared strings, as for the n-gram similarity of Equation 3.10. In other words, Equation 4.8 is a data dependent similarity, unlike all the string similarity measures presented in Chapter 3.

The Fisher kernel of our naive model can be written as:

$$\langle \mathbf{u}(s_i), \mathbf{u}(s_j) \rangle = \sum_{l=1}^{k} \text{sim}_{\text{naive}}(s_i, s^{(l)}) \, \text{sim}_{\text{naive}}(s_j, s^{(l)}) \tag{4.9}$$

This expression is equivalent to the kernel of similarity encoding (Equation 3.13). The important aspect is that, while the similarity metric may vary, the kernel is calculated by summing such a similarity over the set of prototype categories. Such kernels differ from standard string kernels, which capture some form of overlap in substrings between two instances (Leslie et al., 2004; Lodhi et al., 2002). On the contrary, the forms of kernels that appear here create a similarity between two strings that is shaped by the set of reference categories used as prototypes, and their corresponding frequencies.

Part II

SCALABLE CATEGORICAL ENCODERS

5 THE MIN-HASH ENCODING

This chapter presents a stateless encoding method based on hash functions that is suitable for string entries: the *min-hash encoder*. The encoder assigns similar feature vectors to similar string entries in terms of the n-gram similarity between them (Equation 3.10). The stateless property is particularly interesting, as it allows the encoder to be completely independent from the data, enabling its usage in distributed settings. Also, as it is purely based on hash functions, it is very fast to compute. It is also important to remark that the min-hash encoder is the only method we benchmark that is completely independent from the data and still competitive in terms of prediction performance in supervised learning settings (see Chapter 7 for empirical results).

First, we introduce in the next section the *min-hash function*, the building block of the encoder. We also show some of its properties that will help us understand the good performance of the encoder.

5.1 LOCALITY-SENSITIVE HASHING AND THE MIN-HASH.

Locality-sensitive hashing (LSH) (Gionis et al., 1999; Indyk and Motwani, 1998) is a family of hash functions that map similar input items to the same hash code with higher probability than dissimilar items. In this context, hash lookup tables are used in a different way: instead of avoiding collisions, the idea is to put potentially similar items in the same hash bucket. To improve recall, several hash tables can be used. Then, the similarity of two given elements will be proportional to the number of buckets they share.

LSH has been extensively used in Approximate nearest neighbor (ANN) search as an efficient way of finding similar objects (documents, pictures, etc.) in high-dimensional settings. Several hash functions belonging to the LSH family have been proposed depending on the type of distance or similarity that they approximate [1]. For example, the *p-stable distribution* for the ℓ_p norm (Datar et al., 2004), the *sign-random-projection* for cosine similarity (Charikar, 2002), and the *min-hash* for the Jaccard coefficient (Broder, 1997; Broder et al., 2000).

The min-hash is one of the most popular and widely-used functions in the LSH family. It was originally designed to retrieve similar documents in terms of the Jaccard coefficient of their word counts representation (see Leskovec et al., 2014, chapter 3, for a primer). As this type of similarity gives good prediction performance when applied to string categories (Cerda et al., 2018), we will now focus on the analysis of the min-hash function.

1. See Jingdong Wang et al., 2014 for a survey on hashing methods for similarity search and Chi and Zhu, 2017 for a more general survey on hashing techniques.

Let $\mathcal{X}^\star$ be a totally ordered set and π a random permutation of the order in $\mathcal{X}^\star$. For any non-empty $\mathcal{X} \subseteq \mathcal{X}^\star$ with finite cardinality, the min-hash function $Z(\mathcal{X})$ can be defined as:

$$Z(\mathcal{X}) \stackrel{\text{def}}{=} \min_{x \in \mathcal{X}} \pi(x) \tag{5.1}$$

Note that $Z(\mathcal{X})$ can be also seen as a random variable. As shown in Broder, 1997, for any $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{X}^\star$, it is easy to see that the min-hash function has the following property:

$$\mathbb{P}\left(Z(\mathcal{X}){=}Z(\mathcal{Y})\right) = \frac{|\mathcal{X} \cap \mathcal{Y}|}{|\mathcal{X} \cup \mathcal{Y}|} = J(\mathcal{X}, \mathcal{Y}) \tag{5.2}$$

where J is the Jaccard coefficient between the two sets.

MIN-HASH SIGNATURES. For a controlled approximation, several random permutations can be taken, which defines a min-hash signature. For d random permutations $\{\pi_j\}_{j=1}^{d}$ drawn *i.i.d.*, Equation 5.2 leads to:

$$\sum_{j=1}^{d} \mathbb{1}[Z_j(\mathcal{X}) = Z_j(\mathcal{Y})] \sim \mathcal{B}(d, J(\mathcal{X}, \mathcal{Y})) \tag{5.3}$$

where $\mathcal{B}$ denotes the Binomial distribution. Dividing the above quantity by d thus gives a consistent estimate of the Jaccard coefficient $J(\mathcal{X}, \mathcal{Y})$ [2].

Without loss of generality, we can consider the case of $\mathcal{X}^\star$ being equal to the real interval $[0, 1]$, so for any $x \in [0, 1]$, $\pi_j(x) \sim \mathcal{U}(0, 1)$. This allows us to study the min-hash function in terms of probability distributions, as in Proposition 5.1, where we give the cumulative probability distribution of the min-hash.

Proposition 5.1. *Cumulative probability distribution.*

If $\pi(x) \sim \mathcal{U}(0, 1)$, *and* $\mathcal{X} \subset [0, 1]$ *such that* $|\mathcal{X}|{=}k$, *then* $Z(\mathcal{X}) \sim Dir(k, 1)$, *where Dir denotes the Dirichlet distribution* [3].

Proof. It comes directly from considering that:

$$\begin{aligned}
\mathbb{P}(Z(\mathcal{X}) \leqslant z) &= 1 - \mathbb{P}(Z(\mathcal{X}) > z) \\
&= 1 - \prod_{i=1}^{k} \mathbb{P}(\pi(x_i) > z) \\
&= 1 - (1 - z)^k
\end{aligned} \tag{5.4}$$

□

2. Variations of the min-hash signature, as the min-max hash (Ji et al., 2013) have proven to reduce the variance of the Jaccard similarity approximation. The min-max hash uses the min and the max values of the hashed elements, and hence gives two different signature dimensions with only one salt value. We do not study these variations.

3. The Dirichlet distribution $\mathrm{Dir}(\boldsymbol{\alpha})$ with parameter $\boldsymbol{\alpha}{=}(\alpha_1 \dots \alpha_k)$ has as probability density function: $f(\mathbf{x}, \boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^{k} x_i^{\alpha_i - 1}$, with $B(\boldsymbol{\alpha}) = \frac{\prod_{i=1}^{k} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{k} \alpha_i\right)}$.

Now that we know the distribution of the min-hash random variable, we will show how each dimension of a min-hash signature maps inclusion of sets to simple inequalities.

Proposition 5.2. *Inclusion.*

Let $\mathcal{X}, \mathcal{Y} \subset [0,1]$ *such that* $|\mathcal{X}|=k_x$ *and* $|\mathcal{Y}|=k_y$.

(i) If $\mathcal{X} \subset \mathcal{Y}$, *then* $Z(\mathcal{Y}) \leqslant Z(\mathcal{X})$.

(ii) If $\mathcal{X} \subset \mathcal{Y}$, *and* $Z(\mathcal{X}) = z_x$, *then the c.d.f. of* $Z(\mathcal{Y})$ *is:*

$$\mathbb{P}\left(Z(\mathcal{Y}) < z_y\right) = \begin{cases} 1-(1-z_y)^{k_y-k_x} & z_y < z_x \\ 1 & z_x \leqslant z_y \end{cases}$$

(iii) If $\mathcal{X} \cap \mathcal{Y} = \emptyset$, *then:*

$$\mathbb{P}\big(Z(\mathcal{Y}) \leqslant Z(\mathcal{X})\big) = \frac{k_y}{k_x + k_y}$$

Proof. (i) is trivial, *(ii)* is obtained by combining Equation 5.2 and Proposition 5.2.i, and *(iii)* comes directly from Proposition 5.1:

$$\begin{aligned} \mathbb{P}\left(Z(\mathcal{Y}) - Z(\mathcal{X}) \leqslant 0 \,|\, \mathcal{X} \cap \mathcal{Y} = \emptyset\right) &= \int_0^1 \int_0^x f_{Z(\mathcal{Y})}(y) f_{Z(\mathcal{X})}(x)\, dy\, dx \\ &= \int_0^1 \int_0^x k_y (1-x)^{k_y-1} dy\, f_{Z(\mathcal{X})}(x) dx \\ &= \int_0^1 \left(1-(1-x)^{k_y}\right) k_x (1-x)^{k_x-1} dx \\ &= 1 - \int_0^1 k_x (1-x)^{k_y+k_x-1} \\ &= \frac{k_y}{k_x + k_y} \end{aligned} \tag{5.5}$$

□

At this point, we do not know anything about the case when $\mathcal{X} \not\subseteq \mathcal{Y}$, so for a fixed $Z(\mathcal{X})$, we can not ensure that any set with lower min-hash value has $\mathcal{X}$ as inclusion. The following theorem (Cerda and Varoquaux, 2019) allows us to define regions in the vector space generated by the min-hash signature that, with high probability, are associated to inclusion rules.

Theorem 5.1. *Identifiability of inclusion rules.*

Let $\mathcal{X}, \mathcal{Y} \subset [0,1]$ *be two finite sets such that* $|\mathcal{X}|=k_x$ *and* $|\mathcal{Y}|=k_y$.

$$\forall\, \epsilon > 0,\ \textit{if}\ d \geqslant \left\lceil -\frac{\log(\epsilon)}{\log\left(1+\frac{k_x}{k_y}\right)} \right\rceil,\ \textit{then:}$$

$$\mathcal{X} \not\subseteq \mathcal{Y} \Rightarrow \mathbb{P}\left(\sum_{j=1}^{d} \mathbb{1}[Z_j(\mathcal{Y}) \leqslant Z_j(\mathcal{X})] = d\right) \leqslant \epsilon.$$

Proof. First, notice that:

$$\mathcal{X} \not\subseteq \mathcal{Y} \iff \exists\, k \in \mathbb{N}, 0 \leqslant k < k_x \text{ such that } |\mathcal{X} \cap \mathcal{Y}| = k. \tag{5.6}$$

Then, defining $\mathcal{Y}' \stackrel{\text{def}}{=} \mathcal{Y} \setminus (\mathcal{X} \cap \mathcal{Y})$, with $|\mathcal{Y}'| = k_y - k$:

$$\begin{aligned}
\mathbb{P}\left(Z(\mathcal{Y}) \leqslant Z(\mathcal{X}) \,|\, \mathcal{X} \not\subseteq \mathcal{Y}\right) &= \mathbb{P}\left(Z(\mathcal{Y}') \leqslant Z(\mathcal{X}) \,|\, \mathcal{X} \cap \mathcal{Y}' = \emptyset\right) \\
&= \frac{k_y - k}{k_x + k_y - k} \\
&\leqslant \frac{k_y}{k_x + k_y} \\
&= \mathbb{P}\left(Z(\mathcal{Y}) \leqslant Z(\mathcal{X}) \,|\, \mathcal{X} \cap \mathcal{Y} = \emptyset\right)
\end{aligned} \tag{5.7}$$

Finally:

$$\begin{aligned}
\mathbb{P}\left(\sum_{j=1}^{d} \mathbb{1}[Z_j(\mathcal{Y}) \leqslant Z_j(\mathcal{X})] = d \,\middle|\, \mathcal{X} \not\subseteq \mathcal{Y}\right) &= \mathbb{P}\left(Z(\mathcal{Y}) \leqslant Z(\mathcal{X}) \,|\, \mathcal{X} \not\subseteq \mathcal{Y}\right)^d \\
&\leqslant \mathbb{P}\left(Z(\mathcal{X}) \leqslant Z(\mathcal{Y}) \,|\, \mathcal{X} \cap \mathcal{Y} = \emptyset\right)^d \\
&= \left(\frac{k_y}{k_x + k_y}\right)^d
\end{aligned} \tag{5.8}$$

□

Theorem 5.1 tells us that taking enough random permutations ensures that when $\forall j, Z_j(\mathcal{Y}) \leqslant Z_j(\mathcal{X})$, the probability that $\mathcal{X} \not\subseteq \mathcal{Y}$ is small. This result is very important, as it shows a global property of the min-hash representation when using several random permutations, going beyond the well-known properties of collisions in the min-hash signature.

Finally, the next theorem gives us another evidence of the advantages of considering min-hash signatures. In this case, we relate the number of dimensions to the desired Jaccard similarity value γ that one wishes to consider.

Theorem 5.2. *Identifiability of inclusion rules II.*

Let $\mathcal{X}_i \subset [0,1]$ *and* $|\mathcal{X}_i| = k_i < \infty$ *for* $i = 1,2,3$, *such as* $\mathcal{X}_1 \subset \mathcal{X}_3$ *and* $\mathcal{X}_2 \cap \mathcal{X}_3 = \emptyset$. *Suppose also that* $k_1 = k_2 = k$, *and that* $J(\mathcal{X}_1, \mathcal{X}_3) = k_1/k_3 \geqslant \gamma$. *If* $d \geqslant -\log(\epsilon)/\log(1+\gamma)$, *then:*

$$\mathbb{P}\left(\sum_{i=1}^{d} \mathbb{1}[Z_i(\mathcal{X}_2) < Z_i(\mathcal{X}_3)] \geqslant 1\right) \geqslant 1 - \epsilon.$$

Proof.

$$\begin{aligned}
\mathbb{P}\left(Z(\mathcal{X}_2) - Z(\mathcal{X}_3) < 0\right) &= \int_0^1 \int_0^y f_{Z_2}(x)\,dx f_{Z_3}(y)\,dy \\
&= \int_0^1 \int_0^y k_2(1-x)^{k_2-1}\,dx f_{Z_3}(y)\,dy \\
&= \int_0^1 \left(1-(1-y)^{k_2}\right) k_3(1-y)^{k_3-1}\,dy \\
&= 1 - \int_0^1 k_3(1-y)^{k_2+k_3-1} \\
&= \frac{k_2}{k_2+k_3}
\end{aligned} \tag{5.9}$$

As $k_1 = k_2 = k$ and $k_3 \leqslant k/\gamma$, then:

$$\mathbb{P}\left(Z(\mathcal{X}_2) - Z(\mathcal{X}_3) < 0\right) \geqslant \frac{\gamma}{1+\gamma} \tag{5.10}$$

By taking that

$$\sum_{i=1}^{d} \mathbb{1}\left[Z_i(\mathcal{X}_2) < Z_i(\mathcal{X}_3)\right] \sim \mathcal{B}\left(d, \frac{\gamma}{1+\gamma}\right),$$

$$\begin{aligned}\mathbb{P}\left(\sum_{i=1}^{d} \mathbb{1}\left[Z_i(\mathcal{X}_2) < Z_i(\mathcal{X}_3)\right] \geqslant 1\right) &\geqslant 1 - \left(1 - \frac{\gamma}{1+\gamma}\right)^d \\ &= 1 - \left(\frac{1}{1+\gamma}\right)^d \end{aligned} \tag{5.11}$$

□

Theorem 5.2 tells us that, by taking enough min-hash dimensions, with probability $1-\epsilon$ one can find at least one random permutation π_i for which every $Z_i(\mathcal{X}_3)$ such as $\mathcal{X}_3$ satisfies: *(i)* $\mathcal{X}_1 \subset \mathcal{X}_3$, *(ii)* $\mathcal{X}_2 \cap \mathcal{X}_3 = \emptyset$ and *(iii)* $J(\mathcal{X}_1, \mathcal{X}_3) \geqslant \gamma$, can be found in the real range $[Z_i(\mathcal{X}_2), Z_i(\mathcal{X}_1)]$. This is specially relevant for tree based methods, because it facilitates the task of partitioning the space in terms of containment of subword information, independently from the ratio $\frac{k_x}{k_y}$, as in Theorem 5.1.

5.2 THE MIN-HASH ENCODER

A practical way to build a computationally efficient implementation of min-hash is to use a hash function with different salt numbers instead of different random permutations. Indeed, hash functions can be built with desired *i.i.d.* random-process properties (Broder et al., 2000). Thus, the min-hash function can be constructed as follows:

$$Z_j(\mathcal{X}) = \min_{x \in \mathcal{X}} h_j(x), \tag{5.12}$$

where h_j is a hash function[4] on $\mathcal{X}^\star$ with salt value j (also known as seed).

For the specific problem of categorical data, we are interested in a fast approximation of $J(\mathcal{G}(s_i), \mathcal{G}(s_j))$, where $\mathcal{G}(s)$ is the set of all consecutive character n-grams for the string s. For example, the min-hash value of `Paris` is given by:

$$\begin{aligned} Z_j(\mathcal{G}(\texttt{Paris})) &= Z_j(\{\texttt{Par, ari, ris}\}) \\ &= \min\{h_j(\texttt{Par}), h_j(\texttt{ari}), h_j(\texttt{ris})\} \end{aligned}$$

We define the min-hash encoder as:

$$\mathbf{x}^{\text{min-hash}}(s) \stackrel{\text{def}}{=} [Z_1(\mathcal{G}(s)), \ldots, Z_d(\mathcal{G}(s))] \in \mathbb{R}^d. \tag{5.13}$$

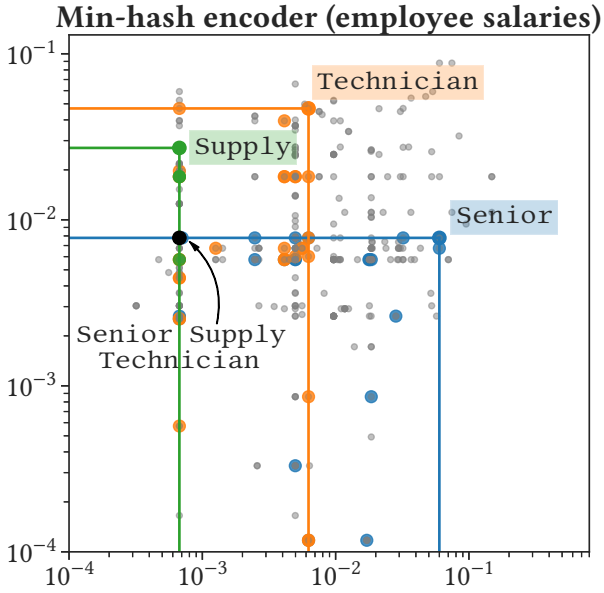


Figure 5.1 – **The min-hash encoder transforms containment into inequality operations.** Color dots are entries in the Employee Salaries dataset (see Section 7.1 in the Appendix) that contain the corresponding colored substrings, and gray dots are categories that do not contain any of them. Given the min-hash properties, the category `Senior Supply Technician` (black dot) must be in the intersection of the three bottom containment regions.

Considering the hash functions as random processes, Equation 5.3 implies that this encoder has the following property:

$$\frac{1}{d}\,\mathbb{E}\left[\|\mathbf{x}^{\text{min-hash}}(s_i) - \mathbf{x}^{\text{min-hash}}(s_j)\|_{\ell_0}\right] = J(\mathcal{G}(s_i), \mathcal{G}(s_j)), \tag{5.14}$$

where $\|\mathbf{x}\|_{\ell_0} = \sum_{i=1}^{d} \mathbb{1}[x_i = 0]$ and $\mathbb{1}[\cdot]$ is the indicator function.

Proposition 5.2 tells us that the min-hash encoder transforms the inclusion relations of strings into an order relation in the feature space. This is especially relevant for learning tree-based models, as theorem 5.1 proves that by performing a small number of splits in the min-hash dimensions, the space can be divided between the elements that contain and do not contain a given substring s.

As an example, Figure 5.1 shows this global property of the min-hash encoder for the case of the Employee Salaries dataset (see Section A.1 in the Appendix) with d=2. The substrings `Senior`, `Supply` and `Technician` are all included in the category formed by concatenating the three words, and as a consequence, the position for this category in the encoding space will be always in the intersection of the bottom-left regions generated by its substrings. For the Employee Salaries dataset, Figure 5.2 illustrates empirically the bound on the dimensionality d and its logarithmic dependence on the desired false positive rate ϵ. It shows that the theoretical bound of the false posi-

4. For the Python implementation of the encoder, we use a 32bit version of the MurmurHash3 function (Appleby, 2014).

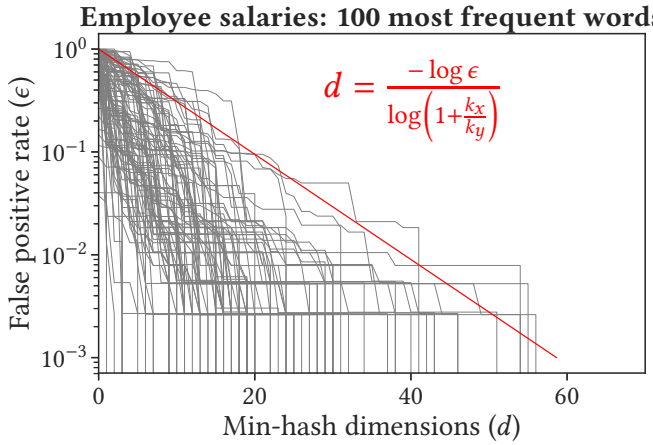


Figure 5.2 – **Number of dimensions required to identify inclusions.** Grey lines are the proportion of false positives obtained for the 100 most frequent words in the employee salaries dataset (H_0 corresponds to identifying categories that do not contain the given word). The red line represents the theoretical minimum dimensionality required to obtain a desired false positive rate (with $k_x/k_y = 0.125$, the inverse of the maximum number of words per category), as shown in Theorem 5.1.

tive rate is respected empirically for 100 of the most frequent words in the Employee Salaries dataset [5].

Finally, algorithm 1 shows the basic steps to compute the min-hash encoder. The encoder is specially suitable for very large scale settings, as it is very fast to compute and completely stateless. A stateless encoding is very useful for distributed computing: different workers can then process data simultaneously without communication. Its drawback is that, as it relies on hashing, the encoding cannot easily be inverted and interpreted in terms of the original string entries.

Algorithm 1: The min-hash encoder

Input : $\{s_i \in \mathcal{S}, i = 1 \dots n\}$ (categorical entries), d (dimensionality of the encoder).

Output : $\mathbf{X} \in \mathbb{R}^{n \times d}$.

Draw s_t from $\{s_1, \dots, s_n\}$.
for $j \in 1 \dots d$ **do**
  Compute $\mathcal{G}(s_t)$
  $x_{tj} \leftarrow \min_{g \in \mathcal{G}(s_t)} h_j(g)$
end

5. The intuition behind this is that identifying the words that compose the string entries (*e.g.*, `Senior Supply Technician`), it is informative for a related learning problem.

6

TOPIC MODELING FOR CATEGORIES: THE GAMMA-POISSON FACTORIZATION

In the NLP literature, *topic modeling* accounts for a family of unsupervised learning techniques that are focused on the discovery of hidden latent structure in text corpora (Alghamdi and Alfalqi, 2015; David M. Blei, 2012; Steyvers and Griffiths, 2007). Usually, the objective is to find low-rank representations of documents, which can be later exploited to perform, for instance, text classification.

To make the link with this literature, we consider in this chapter that the text corpus is, in our case, categorical entries in the training set. As the categorical entries we study are much shorter than text documents and can contain typos, we rely on their substring representation instead: we represent each observation by its count vector of character-level structure of n-grams. Once the low rank representation of categories is estimated, we can consider this as an encoding, and use it in a related prediction problem.

6.1 CHOOSING THE APPROPRIATE TOPIC MODEL

The most important model of the topic modeling family is the Latent Dirichlet allocation (LDA) (David M Blei et al., 2003). The LDA is based on a generative process that explicitly considers the word counts of documents as a linear combination of a reduced number of hidden variables, or *topics*[1]. The estimation of the obtained probability density can then be done by using, for example, variational inference (David M Blei et al., 2017).

Consider an observation of a categorical variable as a string entry described by its n-gram count vector $\mathbf{f} \in \mathbb{N}^m$. As shown by Buntine, 2002, the LDA's generative model that considers d hidden variables can be written as:

$$\mathbf{x} \sim \text{Dirichlet}(\boldsymbol{\alpha}) \tag{6.1}$$

$$\mathbf{f} \,|\, \mathbf{x} \sim \text{Multinomial}(\ell, \mathbf{x}\mathbf{D}) \tag{6.2}$$

where $\mathbf{x} \in \mathbb{R}^d$ are the activations that decompose the observation $\mathbf{f}$ in d prototypes, $\mathbf{D} \in \mathbb{R}^{d \times m}$ is the topic matrix, ℓ is the document length, and $\boldsymbol{\alpha} \in \mathbb{R}_+^{*d}$ is the vector of parameters for the Dirichlet distribution.

1. They idea of using a mixture models for generating text documents has its theoretical fundaments on the de Finetti's representation theorem (De Finetti, 1990). It states that the joint distribution of an infinitely exchangeable sequence of random variables is as if a random parameter were drawn from some distribution and then the random variables in question were independent and identically distributed (David M Blei et al., 2003). The exchangeability of documents it is intrinsic to the probabilistic model and the exchangeability of words is given thanks to the bag-of-words-representation.

A first problem of the LDA (as developed in David M Blei et al., 2003), is that it does not model the document length. As seen in Chapter 2, categorical entries can sometimes be composed of concatenations of several latent categories, so it is important to consider the length of string entries in the generative model.

A simple extension of the original LDA adds the following generative model for the document length ℓ:

$$\lambda \sim \text{Gamma}(\alpha_0, \beta) \tag{6.3}$$

$$\ell \,|\, \lambda \sim \text{Poisson}(\lambda) \tag{6.4}$$

where $\alpha_0 = \sum_{i=1}^{d} \alpha_i$ and β are the shape and scale parameters of the Gamma distribution. This extension comes naturally, as the Poisson distribution is especially appropriate to counting statistics. Following the derivation of Podosinnikova et al., 2015, Appendix B.1, this extra condition for the LDA model is equivalent to considering the following generative model:

$$x_i \sim \text{Gamma}(\alpha_i, \beta_i), \; i = 1 \dots d. \tag{6.5}$$

$$f_j \,|\, (x\Lambda)_j \sim \text{Poisson}((x\Lambda)_j), \; j = 1 \dots m. \tag{6.6}$$

This extension is also known as the *Gamma-Poisson factorization*, a matrix factorization technique developed independently by Canny, 2004, that has been shown to outperform other methods, including the LDA (Canny, 2004; P. K. Gopalan et al., 2014; P. Gopalan et al., 2013). Also, as we will see later in this chapter, the Gamma-Poisson model is easier to compute, as it only needs to be calculated on the non-zero elements of the n-gram counts (P. Gopalan et al., 2013).

In the next section, we present the Gamma-Poisson factorization model in detail and we develop an online algorithm to estimate the latent variables.

6.2 GAMMA-POISSON FACTORIZATION FOR STRING CATEGORIES

To facilitate interpretation of the features generated by a categorical encoder, we now introduce in detail an encoding approach that estimates a decomposition of the string entries in terms of a linear combination of latent categories or *topics*. For this, we rely on the Gamma-Poisson model (Canny, 2004), a matrix factorization technique well-suited to counting statistics. Each categorical entry in its n-gram count representation $\mathbf{f} \in \mathbb{R}^m$ is modeled as a linear combination of d unknown *prototypes* or *topics*, $\boldsymbol{\Lambda} \in \mathbb{R}^{d \times m}$:

$$\mathbf{f} \approx \mathbf{x}\boldsymbol{\Lambda}, \tag{6.7}$$

Here, $\mathbf{x} \in \mathbb{R}^d$ are the activations that decompose the observation $\mathbf{f}$ in the d prototypes contained in $\boldsymbol{\Lambda}$. As we will see later, these prototypes can be seen as latent categories.

Given a training dataset with n samples, the model estimates the unknown prototypes $\boldsymbol{\Lambda}$ by factorizing the data's bag-of-n-grams rep-

resentation $\mathbf{F} \in \mathbb{N}^{n \times m}$, where m is the number of different n-grams in the data:

$$\mathbf{F} \approx \mathbf{X}\boldsymbol{\Lambda}, \tag{6.8}$$

with $\mathbf{X} \in \mathbb{R}^{n \times d}, \boldsymbol{\Lambda} \in \mathbb{R}^{d \times m}$.

As $\mathbf{f}$ is a vector of counts, it is natural to consider a Poisson distribution for each of its elements:

$$p\left(f_j | (\mathbf{x}\boldsymbol{\Lambda})_j\right) = \frac{1}{f_j!}(\mathbf{x}\boldsymbol{\Lambda})_j^{f_j} e^{-(\mathbf{x}\boldsymbol{\Lambda})_j}, \quad j = 1, ..., m. \tag{6.9}$$

For a prior on the elements of $\mathbf{x} \in \mathbb{R}^d$, we use a Gamma distribution, as it is the conjugate prior of the Poisson distribution [2]:

$$p(x_i) = \frac{x_i^{\alpha_i - 1} e^{-x_i/\beta_i}}{\beta_i^{\alpha_i}\,\Gamma(\alpha_i)}, \quad i = 1, ..., d, \tag{6.10}$$

where $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^d$ are the shape and scale parameters of the Gamma distribution for each one of the d topics. The parametrization of the Gamma distribution allows to control the soft sparsity of the estimated activations $\mathbf{X}$ [3].

6.2.1 *Estimation strategy*

To fit the model to the input data, we maximize the likelihood $\mathcal{L}$ of the model, denoted by:

$$\mathcal{L} = \prod_{j=1}^{m} \frac{(\mathbf{x}\boldsymbol{\Lambda})_j^{f_j} e^{-(\mathbf{x}\boldsymbol{\Lambda})_j}}{f_j!} \prod_{i=1}^{d} \frac{x_i^{\alpha_i - 1} e^{-x_i/\beta_i}}{\beta_i^{\alpha_i}\,\Gamma(\alpha_i)} \tag{6.11}$$

And the log-likelihood:

$$\begin{aligned} \log \mathcal{L} = \quad & \sum_{j=1}^{m} f_j \log((\mathbf{x}\boldsymbol{\Lambda})_j) - (\mathbf{x}\boldsymbol{\Lambda})_j - \log(f_j!) + \\ & \sum_{i=1}^{d} (\alpha_i - 1)\log(x_i) - \frac{x_i}{\beta_i} - \alpha_i \log \beta_i - \log \Gamma(\alpha_i) \end{aligned} \tag{6.12}$$

Maximizing the log-likelihood with respect to the parameters of the model $\mathbf{x}$ and $\boldsymbol{\Lambda}$ gives:

$$\frac{\partial}{\partial \Lambda_{ij}} \log \mathcal{L} = \frac{f_j}{(\mathbf{x}\boldsymbol{\Lambda})_j} x_i - x_i \tag{6.13}$$

$$\frac{\partial}{\partial x_i} \log \mathcal{L} = \sum_{j=1}^{m} \frac{f_j}{(\mathbf{x}\boldsymbol{\Lambda})_j} \Lambda_{ij} - \Lambda_{ij} + \frac{\alpha_i - 1}{x_i} - \frac{1}{\beta_i} \tag{6.14}$$

2. In Bayesian statistics, the *conjugate prior* of a likelihood (in this case the Poisson distribution of n-gram counts) is defined as a prior distribution such as the corresponding posterior belongs to the same probability distribution family of the prior.

3. In particular, Canny, 2004 suggest the shape α_i in the range [1.1, 1.4]. We use a shape parameter of 1.1 for all experiments.

As explained in Canny, 2004, these expressions are analogous to solving the following Non-negative matrix factorization (NMF) with the generalized Kullback-Leibler (KL) divergence[4] as loss:

$$\begin{pmatrix} \mathbf{F} \\ \text{Diag}(\boldsymbol{\beta})^{-1} \end{pmatrix} = \mathbf{X} \begin{pmatrix} \boldsymbol{\Lambda} \\ \text{Diag}(\boldsymbol{\alpha}) - \mathrm{I}_d \end{pmatrix} \tag{6.15}$$

The Gamma-Poisson model can be interpreted as a constrained non-negative matrix factorization in which the generalized KL divergence is minimized between $\mathbf{F}$ and $\mathbf{X}\boldsymbol{\Lambda}$, subject to a Gamma prior in the distribution of the elements of $\mathbf{X}$. The Gamma prior induces sparsity in the activations $\mathbf{x}$ of the model. This is a very important property: it allows to express categorical variables in terms of a reduced number of topics, as it is shown in Figure 6.1 for both real and simulated data. The obtained topics are a reflection of the co-occurrences of the n-grams that compose the categorical entries. Figure 6.2 shows the same representation for all entries (gray dots) in the Employee Salaries datasets after a two-dimensional projection with a t-distributed stochastic neighbor embedding (t-SNE) (Maaten and Hinton, 2008). The sparsity of the activations induces a clear clusterisation of entries (see Figure 8.2 for a comparison with other encoding methods).

To solve the NMF problem above, Lee and Seung, 2001, proposes the following recurrences:

$$\Lambda_{ij} \leftarrow \Lambda_{ij} \left(\sum_{\ell=1}^{n} \frac{f_{\ell j}}{(\mathbf{X}\boldsymbol{\Lambda})_{\ell j}} X_{\ell i} \right) \left(\sum_{\ell=1}^{n} X_{\ell i} \right)^{-1} \tag{6.16}$$

$$X_{\ell i} \leftarrow X_{\ell i} \left(\sum_{j=1}^{m} \frac{f_{\ell j}}{(\mathbf{X}\boldsymbol{\Lambda})_{\ell j}} \Lambda_{ij} + \frac{\alpha_i - 1}{X_{\ell i}} \right) \left(\sum_{j=1}^{m} \Lambda_{ij} + \beta_i^{-1} \right)^{-1} \tag{6.17}$$

As $\mathbf{F}$ is a sparse matrix, the summations above only need to be computed on the non-zero elements of $\mathbf{F}$. This fact considerably decreases the computational cost of the algorithm.

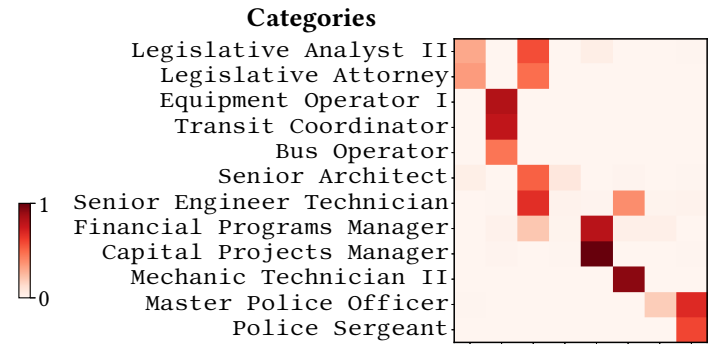
6.2.2 *Online Algorithm*

Following Lefevre et al., 2011, we present an online (or streaming) version of the Gamma-Poisson solver (algorithm 2). The basic idea of the algorithm is to exploit the fact that in the recursion for $\boldsymbol{\Lambda}$ (Equations 6.16 and 6.17), the summations are done with respect to the

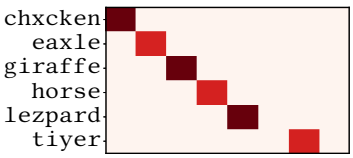
4. In the NMF literature (see for instance Lee and Seung, 2001), the generalized KL divergence for two matrices $\mathbf{X}$ and $\mathbf{Y}$ is defined as:

$$\mathrm{D}(\mathbf{X}\|\mathbf{Y}) = \sum_{i,j} \left(\mathbf{X}_{ij} \log \frac{\mathbf{X}_{ij}}{\mathbf{Y}_{ij}} - \mathbf{X}_{ij} + \mathbf{Y}_{ij} \right).$$

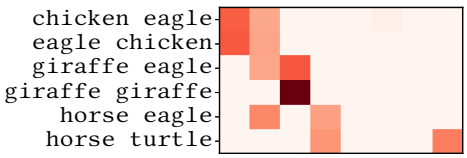
It is called a divergence—and not a distance—because it does not hold the symmetry property in $\mathbf{X}$ and $\mathbf{Y}$.



(a) Employee Salaries dataset (Position Title)



(b) Simulated typos



(c) Simulated multi-label categories

Figure 6.1 – **The Gamma-Poisson factorization gives positive and sparse representations.** Examples of encoding vectors (d=8) for a real dataset (a) and for simulated data (b and c) obtained with a Gamma-Poisson factorization.

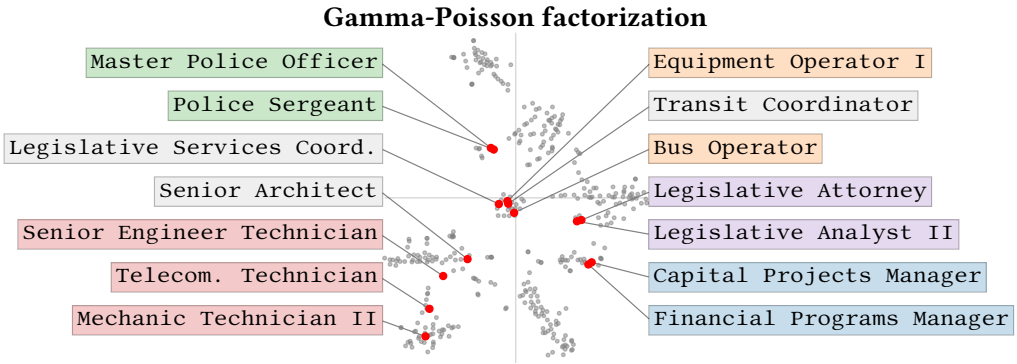


Figure 6.2 – **Projection with t-SNE for the Gamma-Poisson factorization.** The input catagerocial data for this example is the variable *Employee Position Title* in the *Employee Salaries* dataset. The original dimensionality for each encoding is d=10. Gray dots represent the rest of observed categories in the dataset. The Gamma-Poisson creates clusters of categorical entries that can be eventually interpreted.

training samples. Instead of computing the numerator and denominator in the entire training set at each update, one can update this values only with mini-batches of data, which considerably decreases the memory usage and time of the computations.

For better computational performance, we adapt the implementation of this solver to the specificities of our problem—factorizing substring counts across entries of a categorical variable. In particular, we take advantage of the repeated entries by saving a dictionary of the activations for each category in the convergence of the previous mini-batches (algorithm 2, line 4) and use them as an initial guess for the same category in a future mini-batch. This is a warm restart and is

Algorithm 2: Online Gamma-Poisson factorization

Input : $\mathbf{F}$, $\boldsymbol{\Lambda}^{(0)}$, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, ρ, q, η, ϵ
Output : $\mathbf{X}, \boldsymbol{\Lambda}$

1 **while** $\frac{\|\boldsymbol{\Lambda}^{(t)}-\boldsymbol{\Lambda}^{(t-1)}\|_F}{\|\boldsymbol{\Lambda}^{(t-1)}\|_F} > \eta$ **do**
2 draw $\mathbf{f}_t$ from the training set $\mathbf{F}$.
3 **while** $\frac{\|\mathbf{x}_t-\mathbf{x}_t^{old}\|_2}{\|\mathbf{x}_t^{old}\|_2} > \epsilon$ **do**
4 $\mathbf{x}_t \leftarrow \left[\mathbf{x}_t \left(\frac{\mathbf{f}_t}{\mathbf{x}_t\boldsymbol{\Lambda}^{(t)}}\right)\boldsymbol{\Lambda}^{(t)\mathsf{T}} + \boldsymbol{\alpha} - 1\right] . \left[\mathbf{1}\boldsymbol{\Lambda}^{(t)\mathsf{T}} + \boldsymbol{\beta}^{-1}\right]^{\cdot -1}$
5 **end**
6 $\tilde{\mathbf{A}}_t \leftarrow \boldsymbol{\Lambda}^{(t)} . \left[\mathbf{x}_t^{\mathsf{T}} \left(\frac{\mathbf{f}_t}{\mathbf{x}_t\boldsymbol{\Lambda}^{(t)}}\right)\right]$
7 $\tilde{\mathbf{B}}_t \leftarrow \mathbf{x}_t^{\mathsf{T}}\mathbf{1}$
8 **if** $t \equiv 0 \bmod q$, **then**
9 $\mathbf{A}^{(t)} \leftarrow \rho\,\mathbf{A}^{(t-q)} + \sum_{s=t-q+1}^{t} \tilde{\mathbf{A}}^{(s)}$
10 $\mathbf{B}^{(t)} \leftarrow \rho\,\mathbf{B}^{(t-q)} + \sum_{s=t-q+1}^{t} \tilde{\mathbf{B}}^{(s)}$
11 $\boldsymbol{\Lambda}^{(t)} \leftarrow \mathbf{A}^{(t)} ./ \mathbf{B}^{(t)}$
12 **end**
13 $t \leftarrow t+1$
14 **end**

especially important in the case of categorical variables because for most datasets, the number of unique categories is much lower than the number of samples.

6.2.2.1 *Parameter tuning*

The hyper-parameters of the algorithm and its initialization can affect convergence. One important parameter is ρ, the discount factor for the previous iterations of the topic matrix $\boldsymbol{\Lambda}^{(t)}$ (algorithm 2, line 9-10). Figure 6.3 shows that choosing ρ=0.95 gives a good compromise between stability of the convergence and data fitting in term of the generalized KL divergence[5].

With respect to the initialization of the topic matrix $\boldsymbol{\Lambda}^{(0)}$, a good option is to choose the centroids of a k-means clustering (Figure 6.4) in a hashed version of the n-gram count matrix $\mathbf{F}$ (in order to speed-up the k-means algorithm) and then project back to the n-gram space with a nearest neighbors algorithm. A faster alternative, although slightly less performing, is to initialize $\boldsymbol{\Lambda}^{(0)}$ with a *k-means++* algorithm (Arthur and Vassilvitskii, 2007), a seeding procedure for the k-means algorithm in order to guarantee a good convergence. In the case of a streaming setting, both approaches can be used in a subset of the data.

5. The default parameters for the online Gamma-Poisson factorization are listed in Table B.1 in the Appendix. All experiments in Chapter 7 are done with same parametrization.

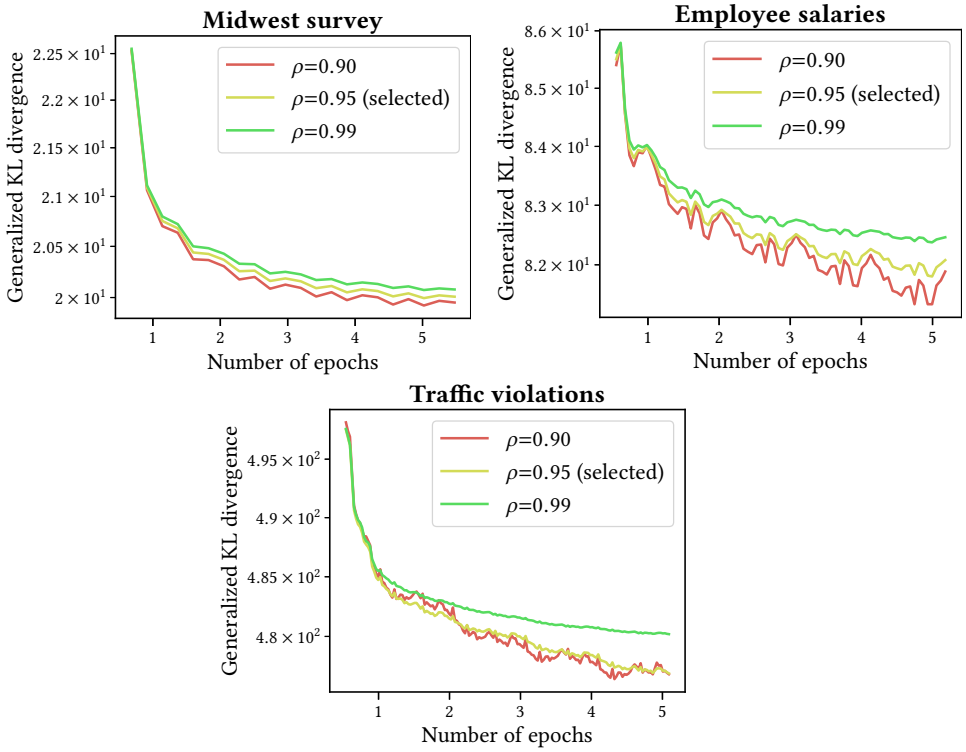


Figure 6.3 – **Convergence for different discount discount values for the Gamma-Poisson model.** The value $\rho = 0.95$ gives a good trade-off between convergence and stability of the solution across the number of epochs.

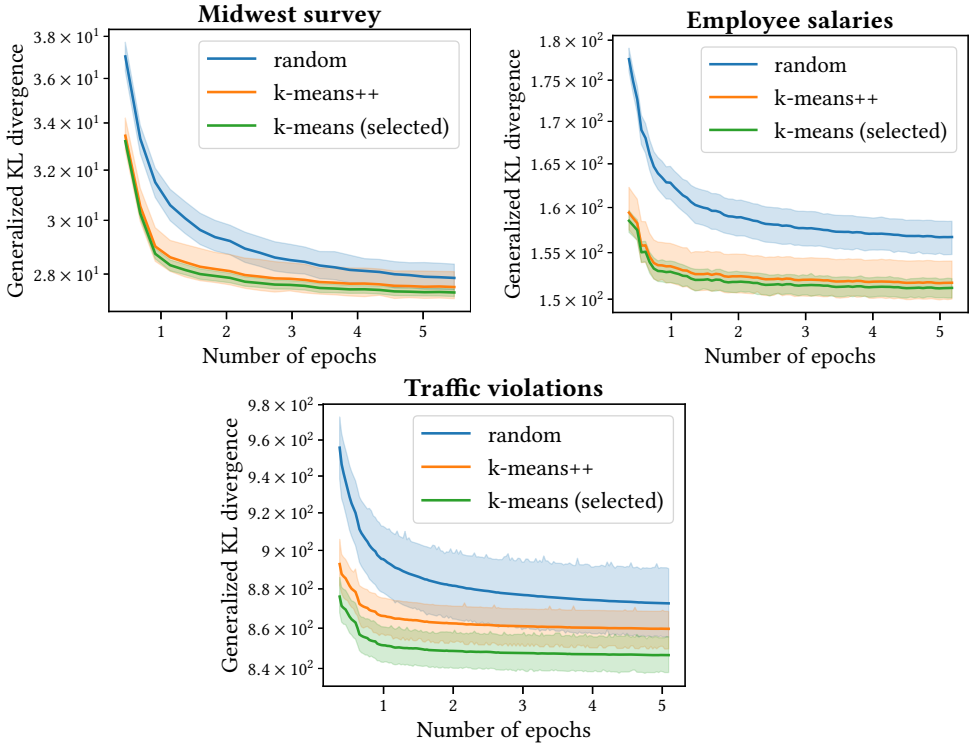


Figure 6.4 – **Convergence for different initialization techniques for the Gamma-Poisson model.** In all benchmark experiments, the k-means strategy is used.

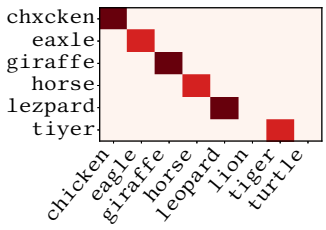
6.3 INFERRING FEATURE NAMES

An encoding strategy where each dimension can be understood by humans facilitates the interpretation of the full statistical analysis. A straightforward strategy for interpretation of the Gamma-Poisson encoder is to describe each encoding dimension by the features of the string entries that it captures. For this, one alternative is to track the feature maps corresponding to each input category, and assign labels based on the input categories that activate the most in a given dimensionality. Another option is to apply the same strategy, but for substrings, such as words contained in the input categories. In the experiments, we follow the second approach as a lot of datasets are composed of entries with overlap, hence individual words carry more information for interpretability than the entire strings.

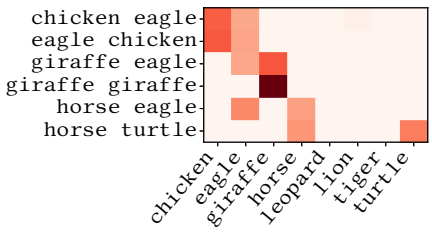
This method can be applied to any encoder, but it is expected to work well if the encodings are sparse and composed only of non-negative values with a meaningful magnitude. The Gamma-Poisson factorization model ensures these properties.

Figure 6.5 shows the same example activations of Figure 6.1, but this time with the corresponding *inferred features names* (x-axis). As the structure of the categorical variable in the real dataset is more complex than the one of the simulations, the feature names in this case are composed by the three most important words for each topic. In Chapter 7 we will see that these feature names can help with the interpretation of a related prediction problem.

(a) Employee Salaries dataset (Occupation)



(b) Simulated typos



(c) Simulated multi-label categories

Figure 6.5 – **The Gamma-Poisson factorization gives positive and sparse representations that are easily interpretable.** Examples of encoding vectors (d=8) for a real dataset (a) and for simulated data (b and c) obtained with a Gamma-Poisson factorization. The x-axis shows the activations for each dimension with their respective inferred feature names. Figure 8.1 shows that other encoders fail to give such an easily-understandable picture.

Part III

EMPIRICAL STUDY

7 SUPERVISED LEARNING BENCHMARKS

This chapter presents a set of benchmarks between different encoding methods in terms of supervised-learning performance. For this purpose, we use real data with curated and non-curated categorical entries, that we describe in detail in the next section.

7.1 REAL-WORLD DATASETS

7.1.1 *Non-curated datasets*

In order to evaluate the different encoding strategies, we collected 17 open source datasets containing a prediction task and at least one relevant high-cardinality categorical variable as feature [1]. Datasets of this type are difficult to collect because most machine learning repositories only contain curated categorical variables (see, for instance, the UCI machine learning repository). To foster future research, all datasets are available for downloading at: https://github.com/dirty-cat/datasets/ [2].

Table 7.1 shows a description of the datasets and the corresponding categorical variables (see Appendix A.1 for more information about the datasets and the related learning tasks). It also details the source of high-cardinality for the respective categorical entries in 4 general non-exclusive types: *multi-label*, *typos*, *description* and *multi-language*. We call *multi-label* the situation when a single column contains multiple information shared by several entries, *e.g.*, `supply technician`, where `supply` denotes the type of activity, and `technician` denotes the rank of the employee (as opposed, *e.g.*, to `supply manager`). *Typos* refers to entries having small morphological variations, as `midwest` and `mid-west`. *Description* refers to categorical entries that are composed of a short free-text description. These are close to a typical NLP problem, although constrained to a very particular subject, so they tend to contain very recurrent informative words and near-duplicate entries. Finally, *multi-language* are datasets in which the categorical variable contains more that one language across the different entries.

7.1.2 *Curated datasets*

We also evaluate the performance of encoders when the categorical entries have already been curated—entries are standardized to create well-defined categorical variables. For this purpose, we collected seven of such datasets (see Section A.1.2 in the Appendix for a

1. If a dataset has more than one categorical variable, only one selected variable was encoded with the proposed approaches, while the rest of them were one-hot encoded.

2. See Section A.1 in the Appendix for a link to the original data sources.

Table 7.1 – **Non-curated datasets.** Description for the corresponding high-cardinality categorical variable. Datasets are ordered by the average number of categories per 1,000 rows (column in bold).

Dataset	#rows	#cat's	#cat's per 1k rows	Gini coeff.	Mean cat. length (#chars)	Source of high cardinality
Crime Data	1.5M	135	**64.5**	0.85	30.6	Multi-label
Medical Charges	163k	100	**99.9**	0.23	41.1	Multi-label
Kickstarter Projects	281k	158	**123.8**	0.64	11.0	Multi-label
Employee Salaries	9.2k	385	**186.3**	0.79	24.9	Multi-label
Open Payments	2.0M	1.4k	**231.9**	0.90	24.7	Multi-label
Traffic Violations	1.2M	11.3k	**243.5**	0.97	62.1	Typos; Description
Vancouver Employees	2.6k	640	**341.8**	0.67	21.5	Multi-label
Federal Election	3.3M	145.3k	**361.7**	0.76	13.0	Typos; Multi-label
Midwest Survey	2.8k	844	**371.9**	0.67	15.0	Typos
Met Objects	469k	26.8k	**386.1**	0.88	12.2	Typos; Multi-label
Drug Directory	120k	17.1k	**641.9**	0.81	31.3	Multi-label
Road Safety	139k	15.8k	**790.1**	0.65	29.0	Multi-label
Public Procurement	352k	28.9k	**804.6**	0.82	46.8	Multi-label-lang.
Journal Influence	3.6k	3.2k	**956.9**	0.10	30.0	Multi-label-lang.
Building Permits	554k	430.6k	**940.0**	0.48	94.0	Typos; Description
Wine Reviews	138k	89.1k	**997.7**	0.23	245.0	Description
Colleges	7.8k	6.9k	**998.0**	0.02	32.1	Multi-label

description of each dataset and link to its respective source). Experiments on these datasets are intended to show the robustness of the encoding approaches to situations where there is no need to reduce the dimensionality of the problem, or when capturing the subword information is not necessarily an issue.

The next section describes the learning pipeline used in most of the benchmarks.

7.2 SUPERVISED LEARNING PIPELINE

We now detail the main aspects of the supervised learning pipeline we built in order to test encoders' prediction performance Note that the same pipeline was used for all datasets; no data-specific intervention was carried out in order to improve the prediction results of a particular dataset.

SAMPLE SIZE. Collected datasets' size range from a couple of thousand to several million samples (see Table 7.1). To reduce computation time on the encoding and learning steps, the number of samples was limited to 100k for large datasets. In these cases, the 100k samples were randomly selected from the available data [3].

3. For the experiments done in Cerda et al., 2018, we limited the number of rows to 10k, as there are benchmarks where the dimensionality of the encoder was not reduced.

Table 7.2 – Score metrics for different prediction problems.

Prediction type	Metric name	Definition
Regression	R^2	$1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$
Binary classif.	Avg. precision	$\sum_{j=1}^{k} p(a_j)\left(r(a_j) - r(a_{j-1})\right)$, with: $p(a) = \frac{\sum_{i=1}^{n} \mathbb{1}[y_i=1]\,\mathbb{1}[\hat{y}_i \geqslant a]}{\sum_{i=1}^{n} \mathbb{1}[\hat{y}_i \geqslant a]}$, $r(a) = \frac{\sum_{i=1}^{n} \mathbb{1}[y_i=1]\,\mathbb{1}[\hat{y}_i \geqslant a]}{\sum_{i=1}^{n} \mathbb{1}[y_i=1]}$, $a \in [0,1]$.
Multi-class	Accuracy	$\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[y_i = \hat{y}_i]$

DATA PREPROCESSING. We removed rows with missing values in the target or in any explanatory variable other than the selected categorical variable, for which we replaced missing entries by the string `nan` and we considered it as an additional category. The only additional preprocessing step for the categorical variable was to transform all entries to lower case.

CROSS-VALIDATION. For every dataset, we made 20 random splits of the data, with one third of samples for testing at each time[4]. In the case of binary and multi-class classification, we performed stratified randomization[5].

PERFORMANCE METRICS. Depending on the type of prediction task, we used different scores to evaluate the performance of the supervised learning problem. Table 7.2 contains a definition of each metric in terms of the class values y_i and the respective predictions $\hat{y}_i$. For *regression*, we used the R^2 *score*, the proportion of variance in the target variable that is explained by the features of the predictive problem. For *binary classification*, with $y_i \in \{0, 1\}$ and $\hat{y}_i \in [0, 1]$, we use the *average precision score*, the area under the precision-recall curve, with respect to the less frequent class (we arbitrarily set the class 1 as the less frequent). Finally, for *multi-class classification*, we use the *accuracy score*, the proportion of correct class predictions. All metrics are upper bounded by 1 (a higher score means a better prediction performance).

4. Experiments in Cerda et al., 2018, were done with 100 random splits and 20% of samples in the test set.

5. In classification problems, the stratified randomization is made with the purpose of preserving the percentage of samples for each class across splits.

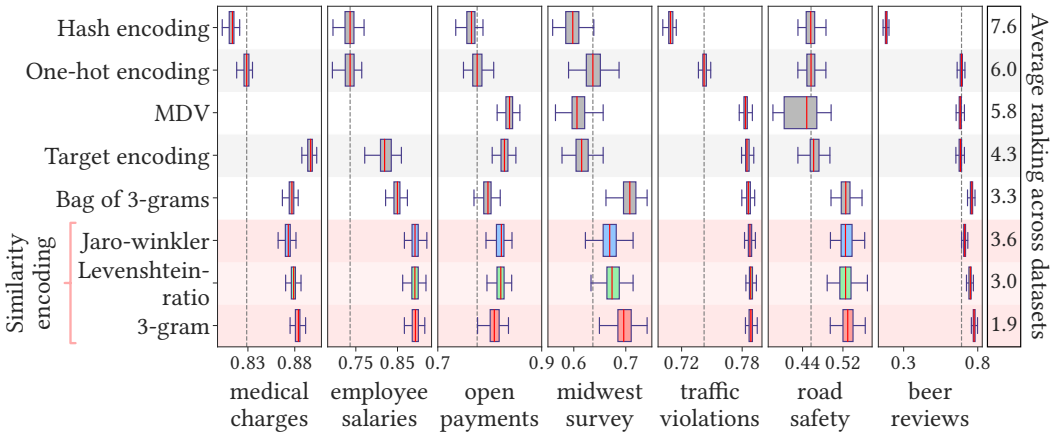


Figure 7.1 – **Performance of different encoding methods.** Classifier: gradient boosting. Each box-plot summarizes the prediction scores of 100 random splits (with 80% of the samples for training and 20% for testing). For all datasets, the prediction score is upper bounded by 1 (a higher score means a better prediction). The right side of the figure indicates the average ranking across datasets for each method. The vertical dashed line indicates the median value of the one-hot encoding method. Similar results are obtained for a linear classifier (see Figure C.1 in the Appendix).

7.3 PREDICTION PERFORMANCE WITH NON-CURATED DATA

In this section we describe the results of several prediction benchmarks with real world non-curated datasets. We start by evaluating the performance of similarity encoding.

7.3.1 *Similarity encoding: choosing a good string similarity*

Similarity encoding (Cerda et al., 2018, Chapter 3) uses continuous string similarity metrics to generalize one-hot encoding. We first benchmark the proposed similarity measures in Section 3.2 (Levenshtein similarity, Jaro-Winkler and n-gram similarity) against the following traditional encoding methods (Chapter 2)[6]:

— One-hot encoding.

— MDV.

— Target Encoding.

— Hash encoding.

— Bag of 3-grams.

Figure 7.1 shows the prediction results (with gradient boosting as classifier) for each method and dataset in terms of the absolute score per dataset (x-axis) and the average ranking across datasets[7]. In average, similarity encoding with n-gram similarity gives the best results.

6. In this section, all encoding strategies based on n-grams are set to use a 3-gram decomposition of strings.

7. At the time of this study, only 7 datasets were available.

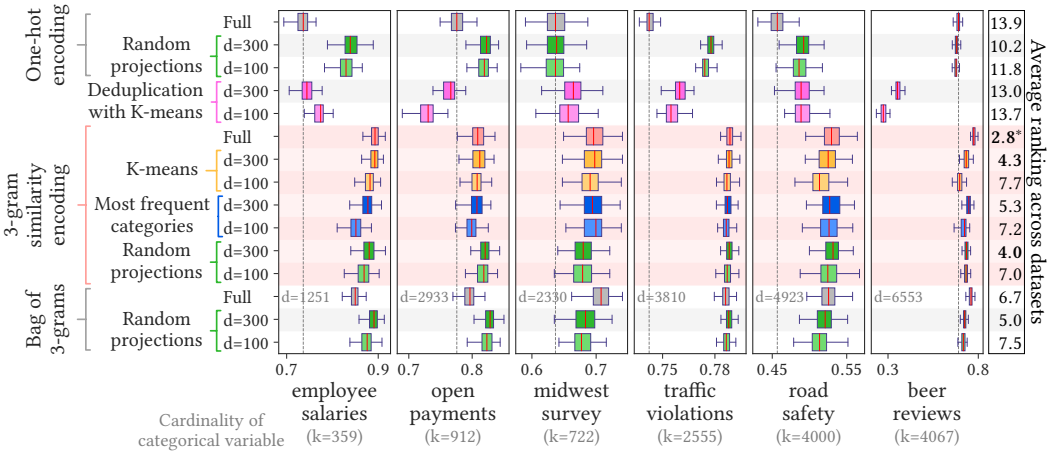


Figure 7.2 – **Prediction performance with different dimensionality reduction strategies**. Classifier: Gradient boosting. *Full* denotes the encoding without dimensionality reduction and d the dimension of the reduction. Each box-plot corresponds to 100 random splits with 80% of the samples for the training set and 20% for the testing set. The right side of the plot indicates the average ranking across datasets for each method (* denotes the best average ranking). Results for a linear classifier are shown in Figure C.2 in the Appendix.

Similar results are obtained for ℓ_2 regularized linear models (see Figure C.1 in the Appendix). Given this results, we selected the n-gram similarity as default similarity metric for all future benchmarks.

In the previous figure, no dimensionality reduction was applied to the encoders. This becomes impracticable in large scale settings. For this reason, the same experiment was done by fixing the maximum dimensionality of encoders. For similarity encoding, we investigated *i)* random projections, *ii)* encoding with similarities to the most frequent categories, *iii)* encoding with similarities to categories closest to the centers of a k-means clustering, and *iv)* one-hot encoding after merging categories with a k-means clustering, which is a simple form of deduplication. The latter method enables bridging the gap with the deduplication literature: we can compare merging entities before statistical learning to expressing their similarity using the same similarity measure. Figure 7.2 shows prediction results for these different dimensionality reduction methods applied to six of the previous seven datasets[8]. Even with a strong dimensionality reduction (d=100), similarity encoding outperforms one-hot encoding.

8. The *Medical Charges* dataset was excluded from the figure because of its smaller cardinality in comparison with the other datasets.

7.3.2 *Benchmarking scalable encoders*

We now describe the results of several prediction benchmarks in 17 non-curated datasets with novel encoders that can scale to large data settings. We mainly benchmark the following encoding strategies [9]:

— One-hot encoding.
— Term-frequency inverse-document-frequency (Tf-idf).
— FastText (Tomas Mikolov et al., 2018).
— Similarity encoding (Chapter 3).
— Min-hash encoding (Chapter 5).
— Gamma-Poisson factorization [10] (Chapter 6).

N-GRAMS RANGE. For all the strategies based on a n-gram representation, we use the set of 2-4 character grams [11], as we noticed that it gives slightly better results than only using the set of 3-grams.

DIMENSIONALITY REDUCTION. Note that one-hot encoding, tf-idf and fastText are naturally high-dimensional encoders, so a dimensionality reduction technique needs to be applied in order to compare the different methodologies. Without this reduction, the benchmark will be unfeasible given the long computational times of gradient boosting. Moreover, dimensionality reduction helps to improve prediction (Cerda et al., 2018) with tree-based methods.

To set the dimensionality of one-hot encoding, Tf-idf and fastText, we used a truncated SVD (implemented efficiently following Halko et al., 2011). We also evaluated Gaussian random projections (Rahimi and Recht, 2008), as it can lead to stateless encoders that require no data fit.

For similarity encoding, we selected prototypes with a *k-means* strategy, following Cerda et al., 2018, as it gives slightly better prediction results than the *most frequent categories*. We do not test the *random projections* strategy for similarity encoding as it is not scalable.

RELATIVE SCORE. As datasets get different prediction scores, we visualize encoders' performance with prediction results scaled in a *relative score*. It is a dataset-specific scaling of the original score, in order to bring performance across datasets in the same range. In other words, for a given dataset i:

$$\text{relative score}_j^i = 100 \frac{\text{score}_j^i - \min_j \text{score}_j^i}{\max_j \text{score}_j^i - \min_j \text{score}_j^i} \tag{7.1}$$

where score_j^i is the the prediction score for the dataset i with the configuration $j \in \mathcal{J}$, the set of all trained models—in terms of dimensionality, type of encoder and cross-validation split. The relative score is

9. Python implementations and example pipelines with these encoding strategies can be found at https://dirty-cat.github.io.
10. Default parameter values are listed in Table B.1
11. In addition to the word as tokens, pretrained versions of fastText also use the set of 3-6 character n-grams.

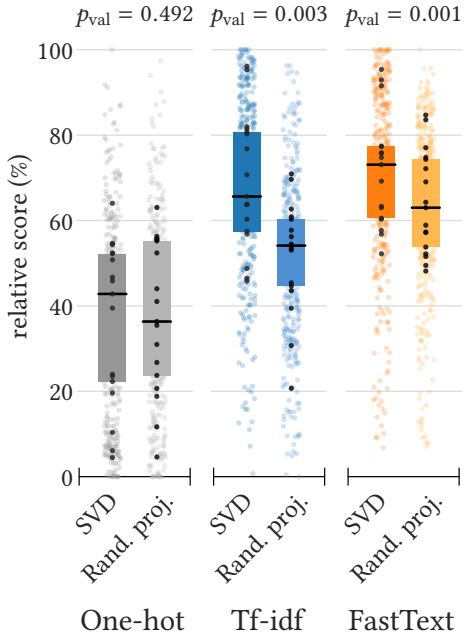


Figure 7.3 – Comparison of encoders' performance with different dimensionality reduction methods (truncated SVD or Gaussian random projection). For each encoder, p_{val} indicates the p-value obtained with a Wilcoxon signed-rank test under the null hypothesis that SVD and random projections have the same performance.

figure-specific and is only intended to be used as a visual comparison of classifiers' performance across multiple datasets. A higher relative score means better results.

STATISTICAL COMPARISON OF ENCODERS. For a proper evaluation of encoders' performance, we use a ranking test across multiple datasets (Demsar, 2006). Note that in such a test each dataset amounts to a single sample, and not the cross-validation splits which are not mutually independent. To do so, for a particular dataset, encoders were ranked according to the median score value over cross-validation splits. At the end, a Friedman test (Friedman, 1937) is used to determine if all encoders, for a fixed dimensionality d, come from the same distribution. If the null hypothesis is rejected, we use a Nemenyi post-hoc test (Nemenyi, 1962) to verify whether the difference in performance across pairs of encoders is significant.

To do pairwise comparison between two encoders, we use a pairwise Wilcoxon signed rank test (Wilcoxon, 1992). The corresponding p-values rejects the null hypothesis that the two encoders are equally performing across different datasets.

CLASSIFIER. For all further experiments, we use gradient boosted trees, as implemented in XGBoost (Chen and Guestrin, 2016). Note that trees can be implemented on categorical variables [12]. However, this encounter the same problems as one-hot encoding: the number

12. XGBoost does not support categorical features. The recommended option is to use one-hot encoding (https://xgboost.readthedocs.io).

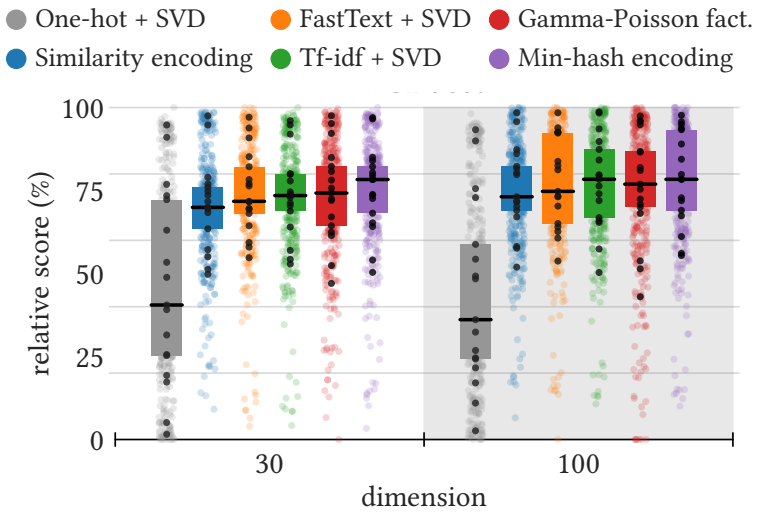


Figure 7.4 – **Encoding with subword information performs significantly better than one-hot.** Classifier: XGBoost. Comparison of encoders in terms of a relative score (the prediction score on the particular dataset, rescaled with respect to the global maximum and minimum score values across dimensions). Color dots indicate the scores for each cross-validation fold, black dots the median score across folds for a dataset, the black line indicates the median score and the box gives the interquartile range. For one-hot encoding, tf-idf, and fastText, the dimensionality was set using an SVD.

of comparisons grows with the number of categories. Hence, the best trees approaches for categorical data use target encoding to impose an order on categories (Prokhorenkova et al., 2018). We also investigated other supervised-learning approaches: linear models, multi-layer perceptron, and kernel machines with RBF and polynomial kernels. However, even with significant hyper-parameter tuning, they under-performed XGBoost on our tabular datasets (Figure C.4 in the Appendix). The good performance of gradient-boosted trees is consistent with previous reports of systematic benchmarks (Olson et al., 2017).

7.3.2.1 *Results*

Figure 7.3 compares the prediction performance of both strategies (for a dimensionality d equal to 30). For tf-idf and fastText, the SVD is significantly superior to random projections. On the contrary, there is no statistical difference for one-hot, but the performance is still slightly superior for the SVD (p-value equal to 0.492). Given these results, we use SVD for all further benchmarks.

Figure 7.4 compares encoders in terms of the relative score of Equation 7.1. All n-gram based encoders clearly improve upon one-hot encoding, at both dimensions (d equal to 30 and 100). Min-hash gives a slightly better prediction performance across datasets, despite of being the only method that does not require a data fit step. Results of the Nemenyi ranking test (Figure 7.5) confirm the impression of the previous visualization: n-gram-based methods are superior to one-

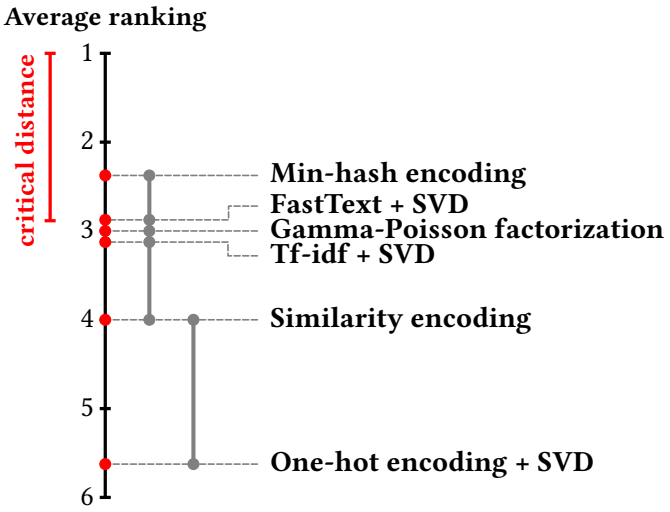


Figure 7.5 – **Nemenyi post-hoc ranking test**. Average ranking across datasets for different encoding methods (d=30). Comparison of encoders against each other with the Nemenyi post-hoc test. Groups of classifiers that are not significantly different (at α=0.05) are connected with a continuous gray line. The red line represents the value of the *critical difference* for rejecting the null hypothesis.

hot encoding, and the min-hash encoder has the best average ranking value for both dimensionalities, although the difference in prediction with respect to the other n-gram based methods is not statistically significant.

FASTTEXT'S LANGUAGE. While we seek generic encoding methods, using precomputed fastText embeddings requires the choice of a lan-

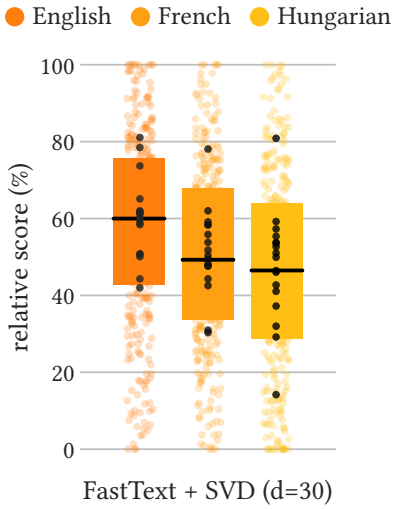


Figure 7.6 – **FastText prediction performance drops for languages other than English.** Relative prediction scores with pretrained fastText vectors in different languages. The dimensionality was set with an SVD. A pairwise Wilcoxon signed rank tests give the following p-values: English-French p=0.056, French-Hungarian p=0.149, English-Hungarian p=0.019.

guage. As 15 out of 17 datasets are fully in English, the benchmarks above use English embeddings for fastText. Figure 7.6, studies the importance of this choice, comparing the prediction results for fastText in different languages (English, French and Hungarian). Not choosing English leads to a sizeable drop in prediction accuracy, which gets bigger for languages more distant (such as Hungarian). This shows that the natural language semantics of fastText indeed are important to explain its good prediction performance. A good encoding not only needs to represent the data in a low dimension, but also needs to capture the similarities between the different entries.

7.4 ROBUSTNESS TO NON CURATED DATA

We now test the robustness of the different encoding methods to situations where there is no need to capture subword information—*e.g.*, low cardinality categorical variables, or variables as "Country name", where the overlap of character n-grams does not have a relevant meaning. We benchmark in Figure 7.7 all encoders on the 7 curated datasets. To simulate black-box usage, the dimensionality was fixed to d=30 for all approaches, with the exception of one-hot encoding. None of the n-gram based encoders perform worst than one-hot. Indeed, the Friedman statistics for the average ranking does not reject the null hypothesis of all encoders coming from the same distribution (p-value equal to 0.37).

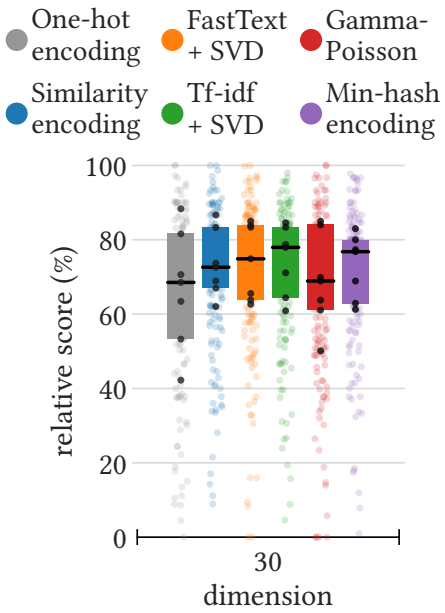


Figure 7.7 – **All encoders perform well for low-cardinality datasets.** Classifier: XGBoost. The score is relative to the best and worse prediction across datasets (Equation 7.1). Color dots indicate the scores for each cross-validation fold, black dots the median across folds, the black line indicates the median across datasets and the box gives the interquartile range. Differences are not significant.

Now that we have shown that the encoding strategies that use the subword information of string entries, we focus on interpretability of the encoders in the next chapter.

8

INTERPRETABLE ANALYSIS ON NON-CURATED CATEGORIES

In the context of machine learning, *interpretability* can be defined as *the ability to explain or to present in understandable terms to a human* (Doshi-Velez and B. Kim, 2017). This ability is critical in many machine learning applications, such as: fairness in decision support systems (Bostrom and Yudkowsky, 2014; Ruggieri et al., 2010), safety in industrial applications (Varshney and Alemzadeh, 2017) or reliability in healthcare (Vellido et al., 2012).

From a machine learning perspective, interpretability can be either *global*, if the concern is to identify the general patterns of a decision function (for instance, the most important features in a trained algorithm), or *local*, if the interest lies in finding the reasons of a specific decision (Doshi-Velez and B. Kim, 2017). In this chapter, we focus in the global interpretability of categorical variables in supervised learning. As the models we consider in this thesis build encoding vectors for categorical variables, we run an empirical study of how well each encoder performs in this respect.

An ideal categorical encoder should have dimensions that can be easily identified and named, as in one-hot encoding, where each dimension corresponds to the presence or absence of a given category. To evaluate this ability in a controlled manner, we created simulated high-cardinality categorical data from a small set of ground truth categories. We describe the data-generation process in the following section.

8.1 SIMULATED CATEGORICAL VARIABLES

In Chapter 7, the description of non-curated datasets in Table 7.1 shows that the most common source for high cardinality string variables is the *multi-label* categories. The second most common source is the presence of *typos* (or any source of morphological variation of the same idea). To analyze these two cases in a more controlled setting, we created two simulated categorical variables. Table 8.1 shows examples of the categories we generated, taking as a base 8 arbitrary ground truth categories of animals: `chicken`, `eagle`, `giraffe`, `horse`, `leopard`, `lion`, `tiger` and `turtle`.

The multi-label data was created by concatenating k+2 ground truth categories, with k following a Poisson distribution—hence, all entries contain at least two labels. For the generation of data with typos, we added 10% of typos to the original ground truth categories by randomly replacing one character by another one (`x`, `y`, or `z`). For both cases, the number of samples was set to 10k.

Table 8.1 – Examples of simulated categorical variables.

Type	Example categories
Ground truth	`chicken`; `eagle`; `giraffe`; `horse`; `leopard`; `lion`; `tiger`; `turtle`.
Multi-label	`lion chicken`; `horse eagle lion`; `tiger leopard giraffe turtle`.
Typos (10%)	`chxcken`; `eazle`; `gixaffe`; `gizaffe`; `hoyse`; `lexpard`; `lezpard`; `lixn`; `tiyer`; `tuxtle`.

8.2 RECOVERING LATENT CATEGORIES

Given the simulated settings, we are interested in measuring the ability of an encoder to recover a feature matrix close to a one-hot encoding matrix of ground-truth categories. For this purpose, we use the Normalized mutual information (NMI) as metric (Witten et al., 2016). Given two random variables X_1 and X_2, the NMI is defined as:

$$\mathrm{NMI} = 2\,\frac{\mathrm{I}(X_1;X_2)}{\mathrm{H}(X_1)+\mathrm{H}(X_2)}, \tag{8.1}$$

where H is the Shannon entropy and I is the mutual information (Shannon, 1948), defined in terms of the entropy as:

$$\mathrm{I}(X_1;X_2) = \mathrm{H}(X_1)+\mathrm{H}(X_2)-\mathrm{H}(X_1,X_2). \tag{8.2}$$

To apply this metric to the feature matrix $\mathbf{X}$ generated by the encoding of all ground truth categories, we consider that $\mathbf{X}$, after rescaling [1], can be seen as a two dimensional probability distribution. For encoders that produce feature matrices with negative values, we take the element-wise absolute value of $\mathbf{X}$. The NMI is a classic measure of correspondences between clustering results (Vinh et al., 2010). Beyond its information-theoretical interpretation, an appealing property is that it is invariant to order permutations. The NMI of any permutation of the identity matrix is equal to 1 and the NMI of any constant matrix is equal to 0. Thus, the NMI in this case is interpreted as a recovering metric of a one-hot encoded matrix of latent, ground truth, categories.

Table 8.2 shows the NMI values for both simulated datasets. The Gamma-Poisson factorization obtains the highest values in both multi-label and typos settings and for different dimensionalities of the encoders. The best recovery is obtained when the dimensionality of the encoder is equal to the number of ground-truth categories, *i.e.*, d=8 in this case.

8.2.1 *Results for real curated data*

For curated data, the cardinality of categorical variables is usually low. Nevertheless, to evaluate how well turn-key generic encodings

1. An ℓ_1 normalization of the rows.

Table 8.2 – **Recovery of categories for simulated categories.** Normalized mutual information (NMI) for different encoders. The data was generated by taking 8 ground truth categories.

Encoder	Multi-label			Typos		
	d=6	**d=8**	d=10	d=6	**d=8**	d=10
Tf-idf + SVD	0.16	0.18	0.17	0.17	0.17	0.17
FastText + SVD	0.08	0.09	0.09	0.08	0.08	0.09
Similarity Encoder	0.32	0.25	0.24	0.72	0.82	0.78
Min-hash Encoder	0.14	0.15	0.13	0.14	0.15	0.13
Gamma-Poisson	**0.76**	**0.82**	**0.79**	**0.78**	**0.83**	**0.80**

Table 8.3 – **Recovering true categories for curated entries.** NMI for different encoders (d=30). Bold values correspond to the highest recovery score by dataset.

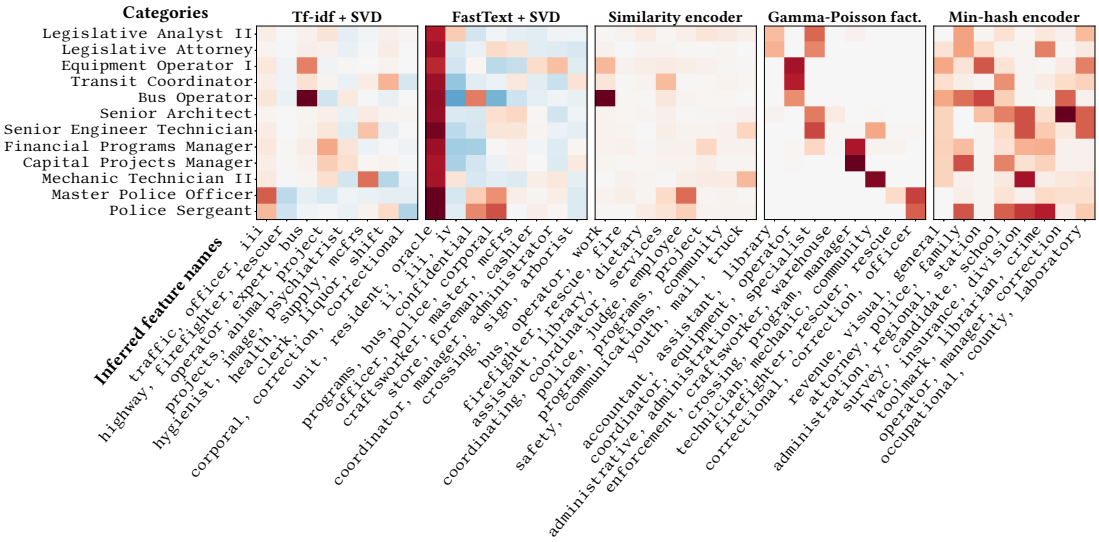
Dataset (cardinality)	Gamma-Poisson	Similarity Encoding	Tf-idf + SVD	FastText + SVD
Adult (15)	**0.75**	0.71	0.54	0.19
Cacao Flavors (100)	**0.51**	0.30	0.28	0.07
California Housing (5)	0.46	0.51	**0.56**	0.20
Dating Profiles (19)	**0.52**	0.24	0.25	0.12
House Prices (15)	**0.83**	0.25	0.32	0.11
House Sales (70)	**0.42**	0.04	0.18	0.06
Intrusion Detection (66)	0.34	**0.58**	0.46	0.11

represent these curated strings, we perform the encoding using a default choice of 30 dimensions. Table 8.3 shows the NMI values for the different curated datasets, without dedicated parameter selection. The NMI measures how much the generated encoding resembles a one-hot encoding on the curated categories. Despite the fact that it is used with a different dimensionality to the cardinality of the curated category, the Gamma-Poisson factorization has the highest recovery performance in 5 out of 7 datasets [2]. This is probably due to the sparsity generated by the Gamma-Poisson factorization (Chapter 6).

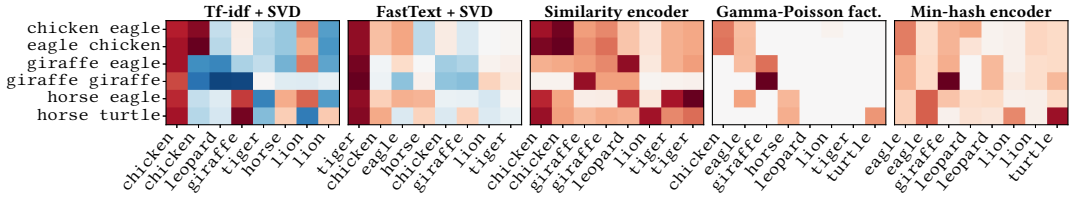
8.3 INTERPRETABILITY OF ENCODERS

As shown in Chapter 6, the Gamma-Poisson factorization creates sparse, non-negative feature vectors that are easily interpretable as a linear combination of latent categories. We give informative features names to each of these latent categories. Experiments on the previous section validate this claim and show that Gamma-Poisson factorization recovers well latent categories.

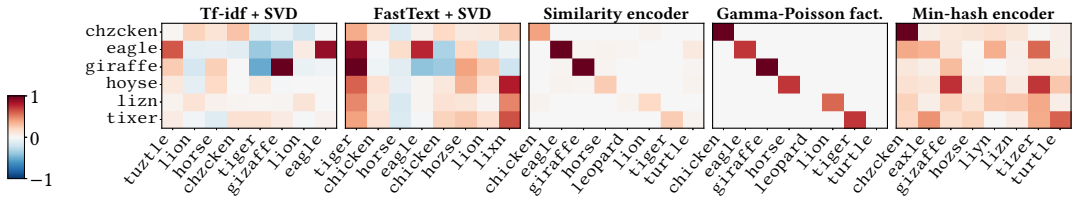
2. Table C.2 in the Appendix show the same analysis but for d=|C|, the actual cardinality of the categorical variable. In this setting, the Gamma-Poisson gives much higher recovery results.



(a) *Employee Position Title* (Employee Salaries dataset)



(b) Simulated multi-label entries



(c) Simulated entries with typos

Figure 8.1 – **The Gamma-Poisson factorization gives positive and sparse representations that are easily interpretable.** Encoding vectors (d=8) for simulated (a and b) and a real dataset (c) obtained with different encoding methods for some categories (y-axis). The x-axis shows the activations with their respective inferred feature names.

To see if other encoding strategies are able to recover ground-truth category in the same way as the Gamma-Poisson Figure 8.1 shows such encodings—and the corresponding inferred feature names—in the case of the simulated data, as well as in the real-world non-curated Employees Salaries dataset. With the exception of similarity encoding for the simulated data with typos, Figure 8.1 confirms that other encodings fail to give relevant feature names with this technique.

Moreover, the sparsity of the Gamma-Poisson factorization allows to visualize clusters of categories with similar substring structure, as it is shown in Figure 8.2, where a 2D t-SNE projection was done for several encoders.

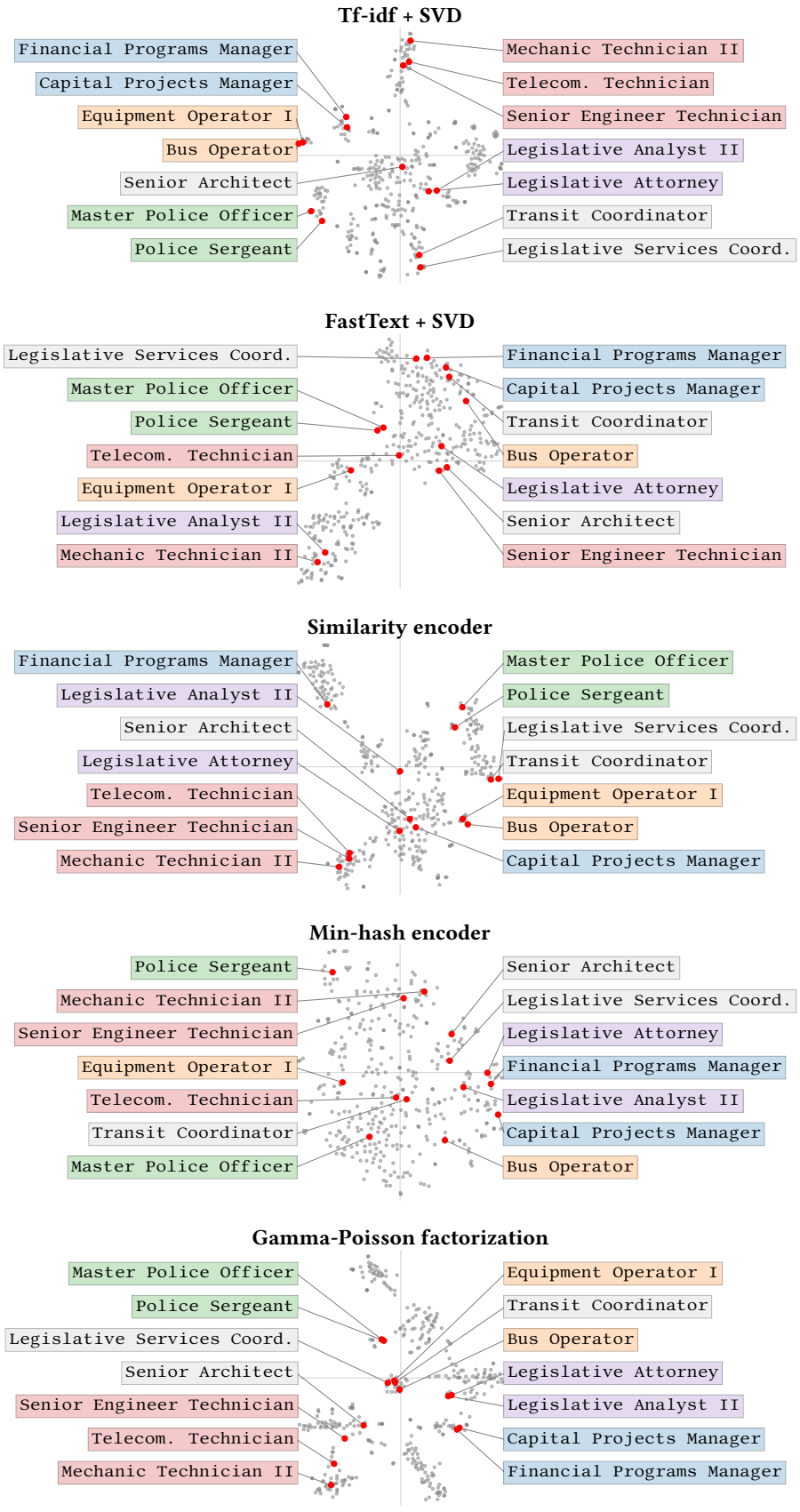


Figure 8.2 – Projection with t-SNE for different encoding methods for the categorical variable *Employee Position Title* in the *Employee Salaries* dataset. The original dimensionality for each encoding is d=10. Gray dots represent the rest of observed categories in the dataset.

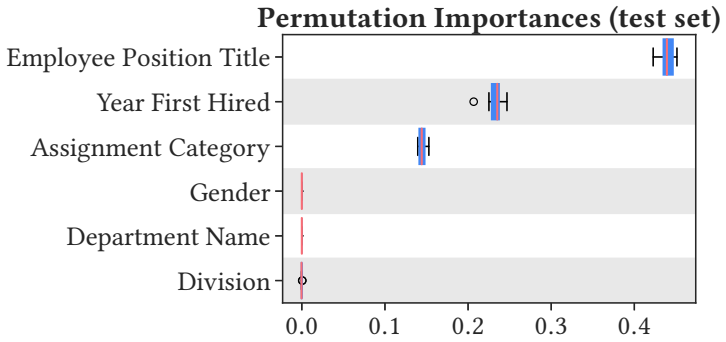


Figure 8.3 – **Overall permutation importances for every feature in the Employee Salaries dataset.** The box plots display permutation importances for every input variable used to predict salaries in the Employee Salaries dataset.

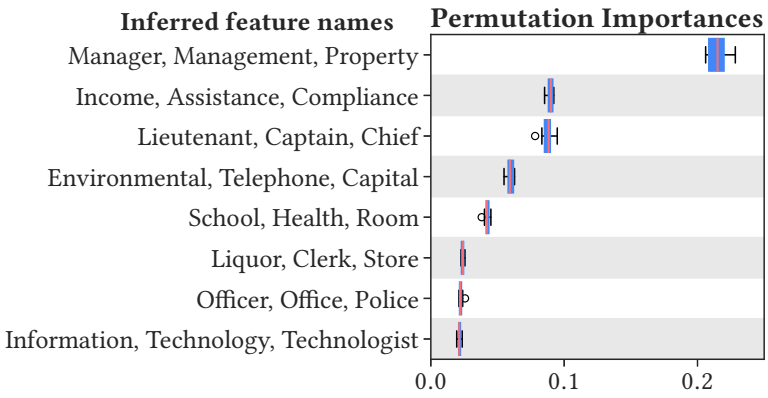


Figure 8.4 – **Gamma-Poisson enables interpretable data science.** The box plots display permutation importances for the variable *Employee Position Title* in the Employee Salaries dataset. Here we show the 8 most important latent topics from a total of 30.

Finally, to illustrate how such encoding can be used in a data-science setting where humans need to understand results, we evaluate the permutation importances (Altmann et al., 2010) of features in the Employee Salaries datasets. First, Figure 8.3 shows the permutation importances for the original features. It shows that the *Employee Position Title* is the most important feature in this particular prediction problem. Then, Figure 8.4 shows the permutation importances of each encoding direction of the Gamma-Poisson factorization and its corresponding inferred feature names. By far, the most important feature name to predict salaries in the Employee Salaries dataset is the latent category `Manager, Management, Property`, which matches general intuitions on salaries, and makes the prediction model understandable by humans.

The results in this chapter show that the Gamma-Poisson factorization's performance is robust to situations with curated and non-curated string categories, and that it can be used to infer informative feature names for the encoding dimensions. This attributes are extremely useful in Automated machine learning (AutoML) settings, as it allows us to use the encoder as a default that is able to give insights about the important features in the problem.

9 CONCLUSION

One-hot encoding is the default method for standard statistical analysis on categorical variables. Beyond its simplicity, its strength is to represent the discrete nature of nominal categories. However, one-hot encoding becomes impractical when there are too many different unique categorical entries.

Encoding categorical variables in non-curated tables has not been studied much in the statistical-learning literature. Yet, it is a common hurdle for data scientists in many application settings. This thesis shows that there is room for improvement upon the standard practice by accounting for similarities in the subword structure across categorical entries. This allows to perform statistical learning on non-curated categorical data without any cleaning or feature engineering step.

To show the benefits of novel encoding approaches, we have collected 17 real-world, non-curated datasets. All of them contain at least one categorical variable with high-cardinality and a related prediction problem. These datasets are openly available, and we hope that they will foster more research on dirty categorical variables. By their diversity, they enable exploring the trade-offs of encoding approaches and concluding on generally-useful defaults. To our knowledge, there is no previous empirical work on benchmarking encoding methods on this topic with such a large data collection.

9.1 GENERALIZING ONE-HOT ENCODING

We start by studying similarity encoding (Chapter 3) [1], a generalization of the one-hot encoding strategy that accounts for the string similarity between categorical entries. Similarity encoding improves the vector representation of categorical variables, especially in the presence of dirty or high-cardinality categorical data.

Experiments show that the n-gram similarity appears to be a good choice to capture morphological resemblance between categories, outperforming similarities typically used for entity resolution such as Jaro-Winkler and Levenshtein. It improves prediction of the associated supervised learning task without any prior data-cleaning step.

While the one-hot representation can be expressed as a sparse matrix, a drawback of similarity encoding is that it creates a dense feature matrix, leading to increased memory and computational costs. However, dimensionality reduction of the resulting matrix maintains most of the benefits of similarity encoding, even with a strong reduction [2]. It greatly reduces the computational cost: fitting the learning

1. A Python implementation is available at https://dirty-cat.github.io/.
2. With Gradient Boosting, similarity encoding reduced to d=30 still outperforms one-hot encoding. Indeed, tree models are good at capturing non-linear decisions in low dimensions.

models on our benchmark datasets takes on the order of seconds or minutes on commodity hardware.

The dimensionality reduction approaches that we have studied can be applied in an online learning setting: they either select a small number prototype categories, or perform a random projection. However, they are based on heuristic rules to select the subset of prototype categories.

From a theoretical standpoint, string similarities appear as an important component of encodings in an analysis with a Fisher kernel (Chapter 4). With a naive generative model of the n-gram vectors of categories, the Fisher kernel induced by the category frequencies leads to encoding new data points as a function of their string similarity to the reference categories. This expression is directly related to the kernel obtained with similarity encoding.

9.2 SCALABLE ENCODERS

In the second part of this thesis, we presented two additional encoding approaches that capture the subword information of string entries. These encoders naturally build low-dimensionality representations that can be applied in online settings and are scalable to large datasets.

We start by proposing the min-hash encoding (Chapter 5), an encoding method based on the idea of Locality-sensitive hashing (LSH) for document retrieval. Besides the well-known local properties of the min-hash signatures, we show that this method transforms string inclusions into inequality operations with high probability.

The min-hash encoder is unique in terms of scalability, as it gives low-dimensional vector representations while being completely stateless. This greatly facilitates distributed computing. The representations enable much better statistical analysis than a simpler random projection of the bag-of-n-grams representations.

Among all strategies studied in this thesis, the min-hash encoding is also the strategy that performs the best in supervised learning tasks, at the cost of some loss in interpretability.

On the other hand, we also studied a matrix factorization technique based on a generative process for the n-gram counts: the Gamma-Poisson factorization (Chapter 6). To make it scalable, we implemented an online algorithm especially optimized for categorical entries. This model naturally captures the subword information of entries by defining hidden topics of interest.

Both models markedly improve upon similarity encoding for large scale learning as: they do not need the definition of a vocabulary of prototype categories; and they naturally give low-dimensional representations, and thus decrease the cost of the subsequent analysis step.

9.3 INTERPRETABILITY

Describing results in terms of a small number of categories can greatly help interpreting a statistical analysis. Our experiments on real and simulated data (Chapter 8) show that encodings created by the Gamma-Poisson factorization correspond to loadings on meaningful recovered categories.

If interpretability of results is an issue, the Gamma-Poisson factorization is a good default encoder for categorical entries. It performs almost as well as the min-hash encoding for supervised learning, and enables expressing results in terms of meaningful latent categories. It removes the need to manually curate entries to understand what drives an analysis. For this, positivity of the loadings and the soft sparsity imposed by the Gamma prior is crucial; a simple SVD fails to give interpretable loadings.

As such, the Gamma-Poisson factorization gives a readily-usable replacement to one-hot encoding for high-cardinality string categorical variables.

9.4 CATEGORICAL ENCODERS FOR AUTOML

AutoML strives to develop machine-learning pipeline that can be applied to datasets without human intervention (Hutter et al., 2015, 2019). To date, it has mainly focused on tuning and model selection for supervised learning on numerical data. Our work addresses the feature-engineering step for string categorical variables. In our experiments, we apply the exact same prediction pipeline to 17 non-curated and 7 curated tabular datasets, without any custom feature engineering. Both Gamma-Poisson factorization and min-hash encoder led to best-performing prediction accuracy, using a classic gradient-boosted tree implementation (XGBoost). We did not tune hyper-parameters of the encoding, such as dimensionality or parameters of the priors for the Gamma-Poisson. These string categorical encodings therefore open the door to AutoML on the original data, removing the need for feature engineering that can lead to difficult model selection. A possible rule when integrating tabular data into an AutoML pipeline could be to apply a min-hash or Gamma-Poisson encoder for string categorical columns with a cardinality above 30, and use one-hot encoding for low-cardinality columns. Indeed, results show that these encoders are also suitable for normalized entries.

APPENDICES

A

REPRODUCIBILITY

A.1 DATASET DESCRIPTION

A.1.1 *Non-curated datasets*

BUILDING PERMITS [1] (sample size: 554k). Permits issued by the Chicago Department of Buildings since 2006. Target (regression): *Estimated Cost*. Categorical variable: *Work Description* (cardinality: 430k).

COLLEGES [2] (7.8k). Information about U.S. colleges and schools. Target (regression): *Percent Pell Grant*. Categorical variable: *School Name* (6.9k).

CRIME DATA [3] (1.5M). Incidents of crime in the City of Los Angeles since 2010. Target (regression): *Victim Age*. Categorical variable: *Crime Code Description* (135).

DRUG DIRECTORY [4] (120k). Product listing data submitted to the U.S. FDA for all unfinished, unapproved drugs. Target (multiclass): *Product Type Name*. Categorical var.: *Non Proprietary Name* (17k).

EMPLOYEE SALARIES [5] (9.2k). Salary information for employees of the Montgomery County, MD. Target (regression): *Current Annual Salary*. Categorical variable: *Employee Position Title* (385).

FEDERAL ELECTION [6] (3.3M). Campaign finance data for the 2011-2012 US election cycle. Target (regression): *Transaction Amount*. Categorical variable: *Memo Text* (17k).

JOURNAL INFLUENCE [7] (3.6k). Scientific journals and the respective influence scores. Target (regression): *Average Cites per Paper*. Categorical variable: *Journal Name* (3.1k).

KICKSTARTER PROJECTS [8] (281k). More than 300,000 projects from https://www.kickstarter.com. Target (binary): *State*. Categorical variable: *Category* (158).

1. https://www.kaggle.com/chicago/chicago-building-permits
2. https://beachpartyserver.azurewebsites.net/VueBigData/DataFiles/Colleges.txt
3. https://data.lacity.org/A-Safe-City/Crime-Data-from-2010-to-Present/y8tr-7khq
4. https://www.fda.gov/Drugs/InformationOnDrugs/ucm142438.htm
5. https://catalog.data.gov/dataset/employee-salaries-2016
6. https://classic.fec.gov/finance/disclosure/ftpdet.shtml
7. https://github.com/FlourishOA/Data
8. https://www.kaggle.com/kemical/kickstarter-projects

MEDICAL CHARGES [9] (163k). Inpatient discharges for Medicare beneficiaries for more than 3,000 U.S. hospitals. Target (regression): *Average Total Payments*. Categorical var.: *Medical Procedure* (100).

MET OBJECTS [10] (469k). Information on artworks objects of the Metropolitan Museum of Art's collection. Target (binary): *Department*. Categorical variable: *Object Name* (26k).

MIDWEST SURVEY [11] (2.8k). Survey to know if people self-identify as Midwesterners. Target (multiclass): *Census Region* (10 classes). Categorical var.: *What would you call the part of the country you live in now?* (844).

OPEN PAYMENTS [12] (2M). Payments given by healthcare manufacturing companies to medical doctors or hospitals (year 2013). Target (binary): *Status* (if the payment was made under a research protocol). Categorical var.: *Company name* (1.4k).

PUBLIC PROCUREMENT [13] (352k). Public procurement data for the European Economic Area, Switzerland, and the Macedonia. Target (regression): *Award Value Euro*. Categorical var.: *CAE Name* (29k).

ROAD SAFETY [14] (139k). Circumstances of personal injury of road accidents in Great Britain from 1979. Target (binary): *Sex of Driver*. Categorical variable: *Car Model* (16k).

TRAFFIC VIOLATIONS [15] (1.2M). Traffic information from electronic violations issued in the Montgomery County, MD. Target (multiclass): *Violation type* (4 classes). Categorical var.: *Description* (11k).

VANCOUVER EMPLOYEE [16](2.6k). Remuneration and expenses for employees earning over $75,000 per year. Target (regression): *Remuneration*. Categorical variable: *Title* (640).

WINE REVIEWS [17] (138k). Wine reviews scrapped from WineEnthusiast. Target (regression): *Points*. Categorical variable: *Description* (89k).

9. https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Inpatient.html
10. https://github.com/metmuseum/openaccess
11. https://github.com/fivethirtyeight/data/tree/master/region-survey
12. https://openpaymentsdata.cms.gov
13. https://data.europa.eu/euodp/en/data/dataset/ted-csv
14. https://data.gov.uk/dataset/road-accidents-safety-data
15. https://catalog.data.gov/dataset/traffic-violations-56dda
16. https://data.vancouver.ca/datacatalogue/employeeRemunerationExpensesOver75k.htm
17. https://www.kaggle.com/zynicide/wine-reviews/home

A.1.2 *Curated datasets*

ADULT [18] (sample size: 32k). Predict whether income exceeds $50K/yr based on census data. Target (binary): *Income*. Categorical variable: *Occupation* (cardinality: 15).

CACAO FLAVORS [19] (1.7k). Expert ratings of over 1,700 individual chocolate bars, along with information on their origin and bean variety. Target (multiclass): *Bean Type*. Categorical variable: *Broad Bean Origin* (100).

CALIFORNIA HOUSING [20] (20k). Based on the 1990 California census data. It contains one row per census block group (a block group typically has a population of 600 to 3,000 people). Target (regression): *Median House Value*. Categorical variable: *Ocean Proximity* (5).

DATING PROFILES [21] (60k). Anonymized data of dating profiles from OkCupid. Target (regression): *Age*. Categorical variable: *Diet* (19).

HOUSE PRICES [22] (1.1k). Contains variables describing residential homes in Ames, Iowa. Target (regression): *Sale Price*. Categorical variable: *MSSubClass* (15).

HOUSE SALES [23] (21k). Sale prices for houses in King County, which includes Seattle. Target (regression): *Price*. Categorical variable: *ZIP code* (70).

INTRUSION DETECTION [24] (493k). Network intrusion simulations with a variaty od descriptors of the attack type. Target (multiclass): *Attack Type*. Categorical variable: *Service* (66).

18. https://archive.ics.uci.edu/ml/datasets/adult
19. https://www.kaggle.com/rtatman/chocolate-bar-ratings
20. https://github.com/ageron/handson-ml/tree/master/datasets/housing
21. https://github.com/rudeboybert/JSE_OkCupid
22. https://www.kaggle.com/c/house-prices-advanced-regression-techniques
23. https://www.kaggle.com/harlfoxem/housesalesprediction
24. https://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data

B

ALGORITHMIC CONSIDERATIONS

B.1 GAMMA-POISSON FACTORIZATION

Algorithm 2 requires some input parameters and initializations that can affect convergence. One important parameter is ρ, the discount factor for the fitting in the past. Figure 6.3 shows that choosing ρ=.95 gives the best compromise between stability of the convergence and data fitting in terms of the Generalized KL divergence.

With respect to the initialization of the topic matrix $\mathbf{\Lambda}^{(0)}$, the best option is to choose the centroids of a k-means clustering (Figure 6.4) in a hashed version of the n-gram count matrix $\mathbf{F}$ with a reduced dimensionality (in order to speed-up convergence of the k-means algorithm) and then project back to the n-gram space with a nearest neighbors algorithm.

The rest of the default values used in the experiments are listed in Table B.1.

Table B.1 – Chosen parametrization for the Gamma-Poisson factorization.

Parameter	Definition	Default value
α_i	Poisson shape	1.1
β_i	Poisson scale	1.0
ρ	Discount factor	0.95
q	Mini-batch size	256
η	Approximation error	10^{-4}
ϵ	Approximation error	10^{-3}

C

ADDITIONAL RESULTS

This chapter shows additional results that are referenced in the main text. They give extra support by extending the main results to a different configuration (a different classifier, encoding or parametrization).

Table C.1 – Median training and encoding times for Gamma-Poisson with XGBoost (d=30).

Datasets	Encoding time Gamma-Poisson	Training time XGBoost	Encoding time / training time
building permits	1522	699	2.18
colleges	17	74	0.24
crime data	28	1910	0.01
drug directory	255	9683	0.03
employee salaries	4	323	0.01
federal election	126	764	0.17
journal influence	7	18	0.37
kickstarter projects	20	264	0.08
medical charge	42	587	0.07
met objects	154	6245	0.02
midwest survey	2	102	0.02
public procurement	547	2150	0.25
road safety	191	1661	0.11
traffic violations	105	1969	0.05
vancouver employee	2	9	0.22
wine reviews	1378	877	1.57

Table C.2 – **Recovering true categories for curated categorical variables.** NMI for different encoders (d=|C|).

Dataset (cardinality)	Gamma-Poisson	Similarity Encoding	Tf-idf + SVD	FastText + SVD
Adult (15)	**0.84**	0.71	0.54	0.19
Cacao Flavors (100)	**0.48**	0.34	0.34	0.1
California Housing (5)	**0.83**	0.51	0.56	0.20
Dating Profiles (19)	**0.47**	0.26	0.29	0.12
House Prices (15)	**0.91**	0.25	0.32	0.11
House Sales (70)	**0.29**	0.03	0.26	0.07
Intrusion Detection (66)	0.27	**0.65**	0.61	0.13

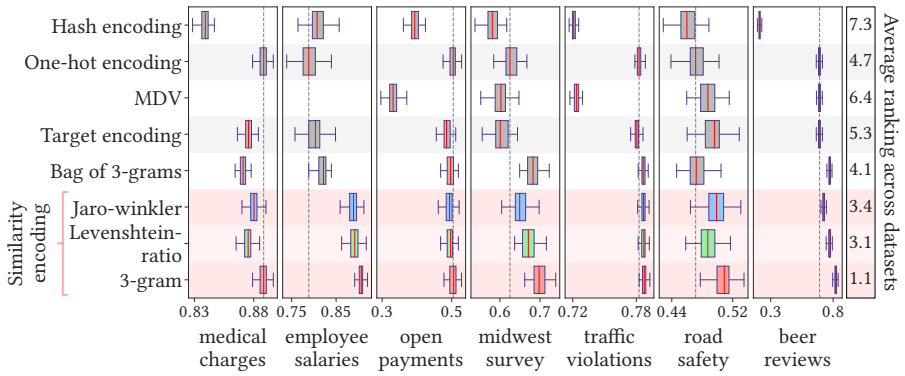


Figure C.1 – **Performance of different encoding methods.** Classifier: Ridge/logistic regression. Each box-plot summarizes the prediction scores of 100 random splits (with 80% of the samples for training and 20% for testing). For all datasets, the prediction score is upper bounded by 1 (a higher score means a better prediction). The right side of the figure indicates the average ranking across datasets for each method. The vertical dashed line indicates the median value of the one-hot encoding method. Similar results are obtained for a linear classifier

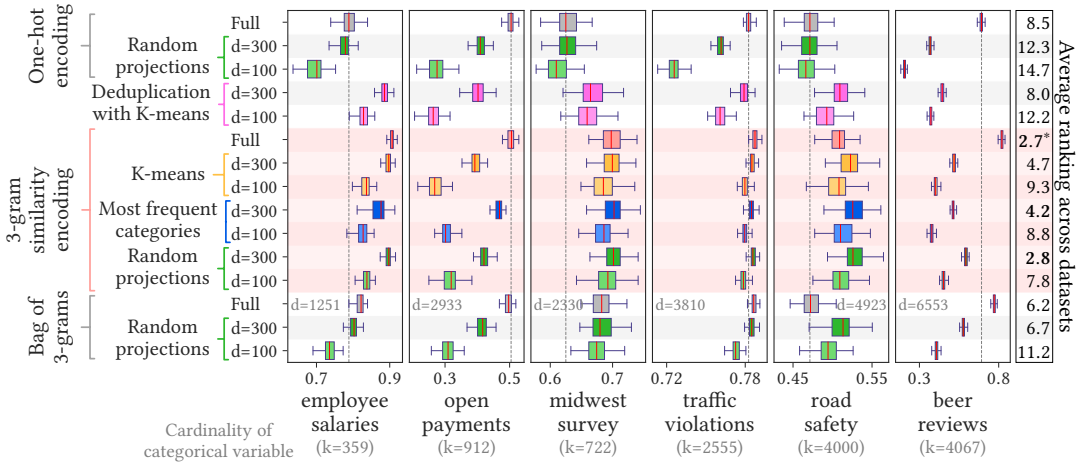
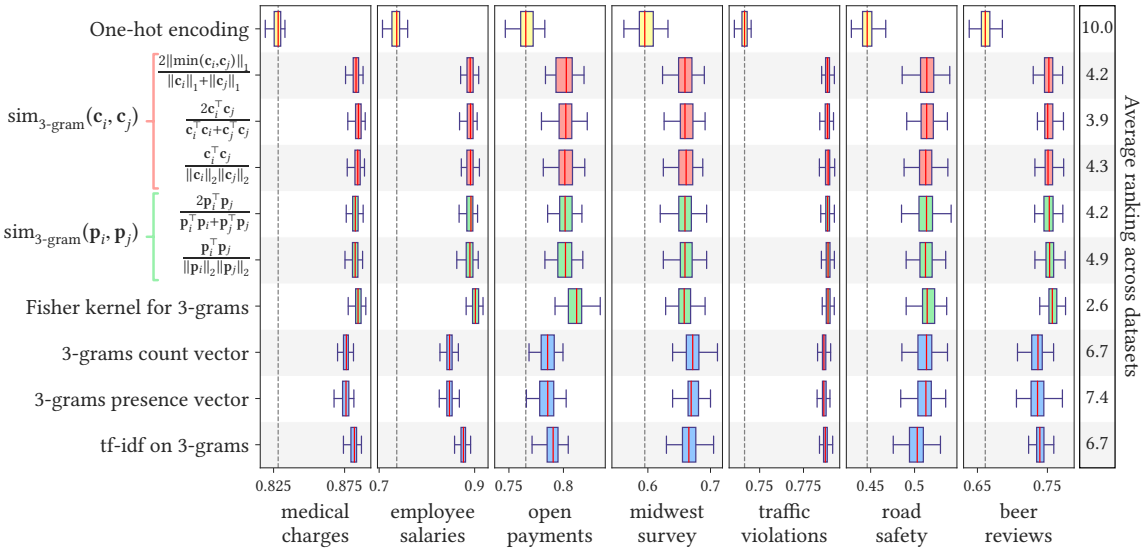
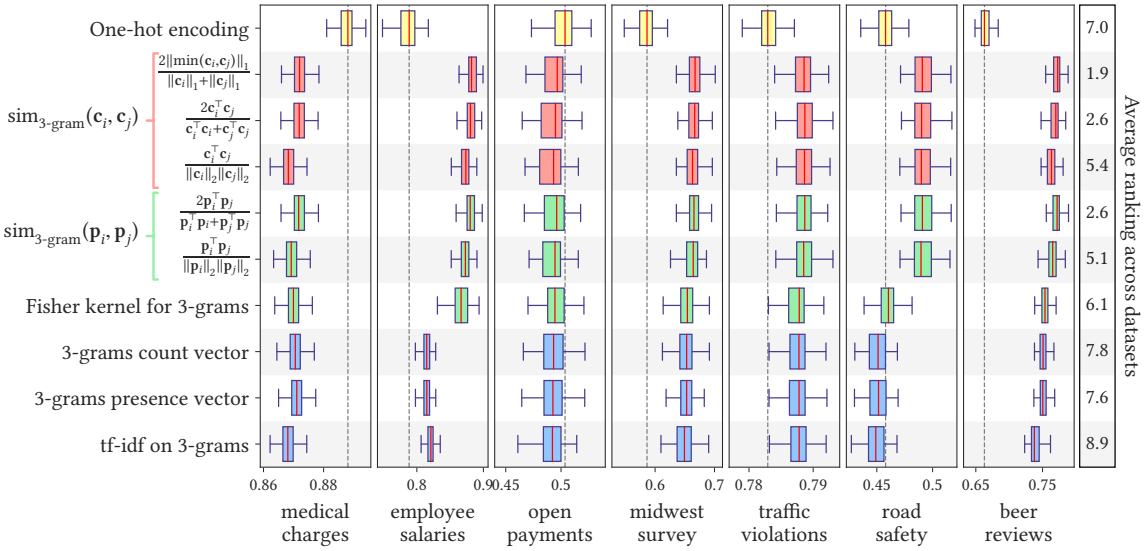


Figure C.2 – **Performance with different dimensionality reduction methods**. Classifier: linear/logistic regression. Full denotes the encoding without dimensionality reduction and d the dimension of the reduction. Each box-plot corresponds to 100 random splits with 80% of the samples for the training set and 20% for the testing set. The right side of the plot indicates the average ranking across datasets for each method (* denotes the best average ranking).

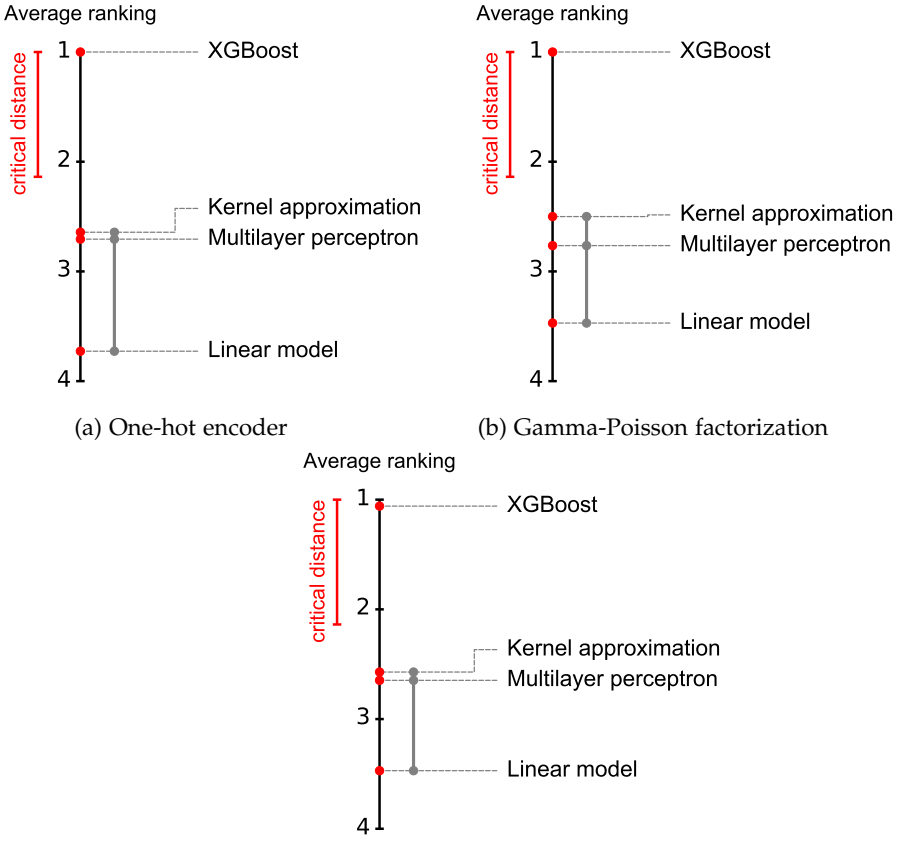


(a) Gradient boosting



(b) Ridge

Figure C.3 – **Performance of different encoding methods.** Upper figure: gradient boosting; Lower figure: ridge. Each boxplot summarizes the prediction scores of 100 random splits (with one third of samples for testing). The methods labels with similarities $\text{sim}_{3\text{-gram}}$, in red and in green, all use an encoding based on similarities as in Cerda et al., 2018. For all datasets, the prediction score is upper bounded by 1 (a higher score means a better prediction). The right side of the figure indicates the average ranking across datasets for each method (a smaller value means a better performance). The vertical dashed line indicates the median value of the one-hot encoding method.



(a) One-hot encoder

(b) Gamma-Poisson factorization

(c) Min-hash encoder

Figure C.4 – **Comparison of classifiers against each other with the Nemenyi post-hoc test** (as in Demsar, 2006). Groups of classifiers that are not significantly different (at α=0.05) are connected with a continuous gray line. The red line represents the value of the *critical difference* for rejecting the null hypothesis. The benchmarked classifiers are: XGBoost; Polynomial kernel approximation with the Nystroem method, followed by an $\ell 2$ regularized linear/logistic regression (kernel approximation); a multilayer perceptron (1-2 layers); and a $\ell 2$ regularized linear/logistic regression (linear model).

REFERENCES

Achlioptas, D. (2003). Database-friendly random projections: johnson-lindenstrauss with binary coins. *Journal of computer and System Sciences*, *66*(4), 671–687.

Agresti, A., & Kateri, M. (2011). *Categorical data analysis*. Springer.

Aldous, D. J. (1985). Exchangeability and related topics. In *École d'Été de Probabilités de Saint-Flour XIII—1983* (pp. 1–198). Springer.

Alghamdi, R., & Alfalqi, K. (2015). A survey of topic modeling in text mining. *Int. J. Adv. Comput. Sci. Appl.(IJACSA)*, *6*(1).

Alkharusi, H. (2012). Categorical variables in regression analysis: a comparison of dummy and effect coding. *International Journal of Education*, *4*(2), 202–210.

Altmann, A., Tolosi, L., Sander, O., & Lengauer, T. (2010). Permutation importance: a corrected feature importance measure. *Bioinformatics*, *26*(10), 1340–1347.

Amari, S., & Nagaoka, H. (2007). *Methods of information geometry*. Amer Mathematical Society.

Angell, R. C., Freund, G. E., & Willett, P. (1983). Automatic spelling correction using a trigram similarity measure. *Information Processing & Management*, *19*(4), 255–261.

Appleby, A. (2014). Murmurhash3 http://code. google. com/p/smhasher/wiki. MurmurHash3.

Arora, S., Liang, Y., & Ma, T. (2016). A simple but tough-to-beat baseline for sentence embeddings.

Arthur, D., & Vassilvitskii, S. (2007). K-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (pp. 1027–1035). Society for Industrial and Applied Mathematics.

Berry, K. J., Mielke Jr, P. W., & Iyer, H. K. (1998). Factorial designs and dummy coding. *Perceptual and motor skills*, *87*(3), 919–927.

Blei, D. M. [David M.]. (2012). Probabilistic topic models. *Commun. ACM*, *55*(4), 77–84.

Blei, D. M. [David M], Kucukelbir, A., & McAuliffe, J. D. (2017). Variational inference: a review for statisticians. *Journal of the American Statistical Association*, *112*(518), 859–877.

Blei, D. M. [David M], Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993–1022.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association of Computational Linguistics*, *5*(1), 135–146.

Bostrom, N., & Yudkowsky, E. (2014). The ethics of artificial intelligence. *The Cambridge handbook of artificial intelligence*, *316*, 334.

Brill, E., & Moore, R. C. (2000). An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual*

Meeting on Association for Computational Linguistics (pp. 286–293). Association for Computational Linguistics.

Broder, A. Z. (1997). On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)* (pp. 21–29). IEEE.

Broder, A. Z., Charikar, M., Frieze, A. M., & Mitzenmacher, M. (2000). Min-wise independent permutations. *Journal of Computer and System Sciences*, *60*(3), 630–659.

Buntine, W. (2002). Variational extensions to em and multinomial pca. In *European Conference on Machine Learning* (pp. 23–34). Springer.

Canny, J. (2004). Gap: a factor model for discrete data. In *ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 122–129).

Cerda, P., & Varoquaux, G. (2019). Encoding high-cardinality string categorical variables. *arXiv preprint arXiv:1907.01860*.

Cerda, P., Varoquaux, G., & Kégl, B. (2018). Similarity encoding for learning with dirty categorical variables. *Machine Learning*.

Charikar, M. S. (2002). Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing* (pp. 380–388). ACM.

Chen, T., & Guestrin, C. (2016). XGBoost: a scalable tree boosting system. In *SIGKDD* (pp. 785–794).

Chi, L., & Zhu, X. (2017). Hashing techniques: a survey and taxonomy. *ACM Computing Surveys (CSUR)*, *50*(1), 11.

Christen, P. (2012a). A survey of indexing techniques for scalable record linkage and deduplication. *IEEE transactions on knowledge and data engineering*, *24*(9), 1537–1555.

Christen, P. (2012b). *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection*. Springer Science & Business Media.

Cohen, P., West, S. G., & Aiken, L. S. (2014). *Applied multiple regression/-correlation analysis for the behavioral sciences*. Psychology Press.

Cohen, W. W., Ravikumar, P., Fienberg, S. E. et al. (2003). A comparison of string distance metrics for name-matching tasks. In *IIWeb* (Vol. 2003, pp. 73–78).

Conrad, C., Ali, N., Keselj, V., & Gao, Q. (2016). Elm: an extended logic matching method on record linkage analysis of disparate databases for profiling data mining. In *2016 IEEE 18th Conference on Business Informatics (CBI)* (Vol. 1, pp. 1–6). IEEE.

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM*, *7*(3), 171–176.

Datar, M., Immorlica, N., Indyk, P., & Mirrokni, V. S. (2004). Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry* (pp. 253–262). ACM.

Davis, M. J. (2010). Contrast coding in multiple regression analysis: strengths, weaknesses, and utility of popular coding structures. *Journal of Data Science*, *8*(1), 61–73.

De Finetti, B. (1990). Theory of probability, vol. i (1974). *John Wiley & Sons*, *5*(8), 17.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, *41*(6), 391–407.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, *7*, 1.

Dheeru, D., & Karra Taniskidou, E. (2017). UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences.

Domingos, P. M. (2012). A few useful things to know about machine learning. *Commun. acm*, *55*(10), 78–87.

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Duch, W., Grudzinski, K., & Stawski, G. (2000). Symbolic features in neural networks. In *In Proceedings of the 5th Conference on Neural Networks and Their Applications*. Citeseer.

Elkan, C. (2005). Deriving tf-idf as a fisher kernel. In *International Symposium on String Processing and Information Retrieval* (pp. 295–300). Springer.

Elmagarmid, A. K., Ipeirotis, P. G., & Verykios, V. S. (2007). Duplicate record detection: a survey. *IEEE Transactions on knowledge and data engineering*, *19*(1), 1–16.

Eskin, E., Weston, J., Noble, W. S., & Leslie, C. S. (2003). Mismatch string kernels for SVM protein classification. In *Advances in neural information processing systems* (pp. 1441–1448).

Fellegi, I. P., & Sunter, A. B. (1969). A theory for record linkage. *Journal of the American Statistical Association*, *64*(328), 1183–1210.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, *32*(200), 675–701.

Gärtner, T. (2003). A survey of kernels for structured data. *ACM SIGKDD Explorations Newsletter*, *5*(1), 49–58.

Gionis, A., Indyk, P., Motwani, R. et al. (1999). Similarity search in high dimensions via hashing. In *Vldb* (Vol. 99, *6*, pp. 518–529).

Gomaa, W. H., & Fahmy, A. A. (2013). A survey of text similarity approaches. *International Journal of Computer Applications*, *68*(13), 13–18.

Gopalan, P. K., Charlin, L., & Blei, D. (2014). Content-based recommendations with poisson factorization. In *Advances in Neural Information Processing Systems* (pp. 3176–3184).

Gopalan, P., Hofman, J. M., & Blei, D. M. (2013). Scalable recommendation with poisson factorization. *arXiv preprint arXiv:1311.1704*.

Grabczewski, K., & Jankowski, N. (2003). Transformations of symbolic data for continuous data oriented models. In *Artificial Neural Networks and Neural Information Processing* (pp. 359–366). Springer.

Guo, C., & Berkhahn, F. (2016). Entity embeddings of categorical variables. *arXiv preprint arXiv:1604.06737*.

Halko, N., Martinsson, P.-G., & Tropp, J. (2011). Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, *53*, 217.

Hamming, R. W. (1950). Error detecting and error correcting codes. *The Bell system technical journal*, *29*(2), 147–160.

Hastie, T., Tibshirani, R., Friedman, J., & Franklin, J. (2005). The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, *27*(2), 83–85.

Hull, D. A. et al. (1996). Stemming algorithms: a case study for detailed evaluation. *JASIS*, *47*(1), 70–84.

Hutter, F., Kegl, B., Caruana, R., Guyon, I., Larochelle, H., & Viegas, E. (2015). Automatic machine learning (automl). In *ICML Workshop on Resource-Efficient Machine Learning*.

Hutter, F., Kotthoff, L., & Vanschoren, J. (2019). *Automated machine learning-methods, systems, challenges*. Springer.

Indyk, P., & Motwani, R. (1998). Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing* (pp. 604–613). ACM.

Jaakkola, T., & Haussler, D. (1999). Exploiting generative models in discriminative classifiers. In *Advances in neural information processing systems* (pp. 487–493).

Jaakkola, T. S., Diekhans, M., & Haussler, D. (1999). Using the fisher kernel method to detect remote protein homologies. In *ISMB* (Vol. 99, pp. 149–158).

Jaro, M. A. (1989). Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, *84*(406), 414–420.

Ji, J., Li, J., Yan, S., Tian, Q., & Zhang, B. (2013). Min-max hash for jaccard similarity. In *2013 IEEE 13th International Conference on Data Mining* (pp. 301–309). IEEE.

Johnson, W. B., & Lindenstrauss, J. (1984). Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, *26*(189-206), 1.

Kim, W., Choi, B.-J., Hong, E.-K., Kim, S.-K., & Lee, D. (2003). A taxonomy of dirty data. *Data mining and knowledge discovery*, *7*(1), 81–99.

Klein, D., Smarr, J., Nguyen, H., & Manning, C. D. (2003). Named entity recognition with character-level models. In *conference on Natural language learning at HLT-NAACL* (p. 180).

Kondrak, G. (2005). N-gram similarity and distance. In *International symposium on string processing and information retrieval* (pp. 115–126). Springer.

Landauer, T. K., Foltz, P. W., & Laham, D. (1998). An introduction to latent semantic analysis. *Discourse processes*, *25*, 259.

Lee, D. D., & Seung, H. S. (2001). Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems* (pp. 556–562).

Lefevre, A., Bach, F., & Févotte, C. (2011). Online algorithms for nonnegative matrix factorization with the itakura-saito divergence.

In *Applications of Signal Processing to Audio and Acoustics (WASPAA)* (p. 313). IEEE.

Leskovec, J., Rajaraman, A., & Ullman, J. D. (2014). *Mining of massive datasets*. Cambridge university press.

Leslie, C. S., Eskin, E., Cohen, A., Weston, J., & Noble, W. S. (2004). Mismatch string kernels for discriminative protein classification. *Bioinformatics*, *20*(4), 467–476.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady* (Vol. 10, *8*, pp. 707–710).

Likert, R. (1932). A technique for the measurement of attitudes. *Archives of psychology*.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2002). Text classification using string kernels. *Journal of Machine Learning Research*, *2*(Feb), 419–444.

Lovins, J. B. (1968). Development of a stemming algorithm. *Mech. Translat. & Comp. Linguistics*, *11*(1-2), 22–31.

Maaten, L. v. d., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, *9*(Nov), 2579–2605.

Maier, D. (1983). *The theory of relational databases*. Computer science press Rockville.

Micci-Barreca, D. (2001). A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. *ACM SIGKDD Explorations Newsletter*, *3*(1), 27–32.

Mikolov, T. [T.], Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. In *ICLR*.

Mikolov, T. [Tomas], Grave, E., Bojanowski, P., Puhrsch, C., & Joulin, A. (2018). Advances in pre-training distributed word representations. In *International Conference on Language Resources and Evaluation (LREC)*.

Moreno, P. J., & Rifkin, R. (2000). Using the fisher kernel method for web audio classification. In *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100)* (Vol. 4, pp. 2417–2420). IEEE.

Myers, J. L., Well, A., & Lorch, R. F. (2010). *Research design and statistical analysis*. Routledge.

Navarro, G. (2001). A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, *33*(1), 31–88.

Nemenyi, P. (1962). Distribution-free multiple comparisons. In *Biometrics* (Vol. 18, p. 263).

O'Grady, K. E., & Medoff, D. R. (1988). Categorical variables in multiple regression: some cautions. *Multivariate behavioral research*, *23*(2), 243–2060.

Oliveira, P., Rodrigues, F., & Henriques, P. R. (2005). A formal definition of data quality problems. In *Proceedings of the 2005 International Conference on Information Quality (MIT IQ Conference)*.

Olson, R. S., La Cava, W., Mustahsan, Z., Varik, A., & Moore, J. H. (2017). Data-driven advice for applying machine learning to bioinformatics problems. *arXiv preprint arXiv:1708.05070*.

Pedhazur, E. J., Kerlinger, F. N. et al. (1973). *Multiple regression in behavioral research*. Holt, Rinehart and Winston New York.

Pennington, J., Socher, R., & Manning, C. (2014). Glove: global vectors for word representation. In *EMNLP* (pp. 1532–1543).

Perronnin, F., & Dance, C. (2007). Fisher kernels on visual vocabularies for image categorization. In *2007 IEEE conference on computer vision and pattern recognition* (pp. 1–8). IEEE.

Podosinnikova, A., Bach, F., & Lacoste-Julien, S. (2015). Rethinking lda: moment matching for discrete ica. In *Advances in Neural Information Processing Systems* (pp. 514–522).

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A., & Gulin, A. (2018). Catboost: unbiased boosting with categorical features. In *Neural Information Processing Systems* (p. 6639).

Pyle, D. (1999). *Data preparation for data mining*. Morgan Kaufmann.

Rahimi, A., & Recht, B. (2008). Random features for large-scale kernel machines. In *Neural Information Processing Systems* (p. 1177).

Rahm, E., & Do, H. H. (2000). Data cleaning: problems and current approaches. *IEEE Data Engineering Bulletin*, *23*, 3.

Ruggieri, S., Pedreschi, D., & Turini, F. (2010). Data mining for discrimination discovery. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *4*(2), 9.

Salton, G., & McGill, M. J. (1983). *Introduction to modern information retrieval*. mcgraw-hill.

Sarawagi, S., & Bhamidipaty, A. (2002). Interactive deduplication using active learning. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 269–278). ACM.

Schölkopf, B., & Smola, A. J. (1998). *Learning with kernels*. MIT Press.

Serva, M., & Petroni, F. (2008). Indo-european languages tree by levenshtein distance. *EPL (Europhysics Letters)*, *81*(6), 68005.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell system technical journal*, *27*(3), 379–423.

Steyvers, M., & Griffiths, T. (2007). Probabilistic topic models. *Handbook of latent semantic analysis*, *427*(7), 424–440.

Ukkonen, E. (1993). Approximate string-matching over suffix trees. In *Annual Symposium on Combinatorial Pattern Matching* (pp. 228–242). Springer.

Vapnik, V. (2013). *The nature of statistical learning theory*. Springer science & business media.

Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE transactions on neural networks*, *10*(5), 988–999.

Varshney, K. R., & Alemzadeh, H. (2017). On the safety of machine learning: cyber-physical systems, decision sciences, and data products. *Big data*, *5*(3), 246–255.

Vellido, A., Martin-Guerrero, J. D., & Lisboa, P. J. (2012). Making machine learning models interpretable. In *ESANN* (Vol. 12, pp. 163–172). Citeseer.

Vinh, N. X., Epps, J., & Bailey, J. (2010). Information theoretic measures for clusterings comparison: variants, properties, normal-

ization and correction for chance. *Journal of Machine Learning Research*, *11*(Oct), 2837–2854.

Wang, H., & Wang, J. [Jingbin]. (2014). An effective image representation method using kernel classification. In *2014 IEEE 26th international conference on tools with artificial intelligence* (pp. 853–858). IEEE.

Wang, J. [Jingdong], Shen, H. T., Song, J., & Ji, J. (2014). Hashing for similarity search: a survey. *arXiv preprint arXiv:1408.2927*.

Weinberger, K., Dasgupta, A., Langford, J., Smola, A., & Attenberg, J. (2009). Feature hashing for large scale multitask learning. In *ICML* (p. 1113). ACM.

Wilcoxon, F. (1992). Individual comparisons by ranking methods. In *Breakthroughs in statistics* (pp. 196–202). Springer.

Winkler, W. E. (1999). The state of record linkage and current research problems. In *Statistical Research Division, US Census Bureau*. Citeseer.

Winkler, W. E. (2002). *Methods for record linkage and bayesian networks*. Technical report, Statistical Research Division, US Census Bureau, Washington, DC.

Winkler, W. E. (2006). Overview of record linkage and current research directions. In *Bureau of the Census*. Citeseer.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data mining: practical machine learning tools and techniques*. Morgan Kaufmann.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Apprentissage statistique à partir de variables catégorielles non-uniformisées

Mots clés : Apprentissage statistique, variables catégorielles, données sales, données volumineuses, autoML.

Résumé : Les données de type tabulaire contiennent souvent des *variables catégorielles*, considérées comme des entrées non numériques avec un nombre fixe et limité d'éléments uniques, appelés *catégories*. De nombreux algorithmes d'apprentissage statistique nécessitent une représentation numérique des variables catégorielles. Une étape d'encodage est donc nécessaire pour transformer ces entrées en vecteurs. Pour cela, plusieurs stratégies existent, dont la plus courante est celle de *l'encodage one-hot*, qui fonctionne bien dans le cadre de l'analyse statistique classique (en termes de puissance de prédiction et d'interprétation) lorsque le nombre de catégories reste faible.
Cependant, les données catégorielles non-uniformisées présentent le risque d'avoir une grande cardinalité et des redondances. En effet, les entrées peuvent partager des informations sémantiques et/ou morphologiques, et par conséquent, plusieurs entrées peuvent refléter la même entité. Sans une étape de nettoyage ou d'agrégation au préalable, les méthodes d'encodage courantes peuvent perdre en efficacité du fait d'une représentation vectorielle erronée. En outre, le risque d'obtenir des vecteurs de très grandes dimensions croit avec la quantité de données, ce qui empêche leur utilisation dans l'analyse de données volumineuses.
Dans ce document, nous étudions une série de méthodes d'encodage qui permettent de travailler directement sur des variables catégorielles à grande cardinalité, sans qu'il soit nécessaire de les traiter en amont. A l'aide d'expériences menées sur des données réelles et simulées, nous démontrons que les méthodes proposées dans le cadre de cette thèse améliorent l'apprentissage supervisé et ce, en autre, du fait de leur capacité à capturer correctement l'information morphologique des entrées. Même avec des données volumineuses, ces méthodes s'avèrent être performantes, et dans certains cas, elles génèrent des vecteurs facilement interprétables. Par conséquent, nos méthodes peuvent être appliquées à l'apprentissage statistique automatique (AutoML) sans aucune intervention humaine.

Title : Statistical learning with high-cardinality string categorical variables

Keywords : Statistical learning, string categorical variables, large-scale data, autoML.

Abstract : Tabular data often contain columns with *categorical variables*, usually considered as non-numerical entries with a fixed and limited number of unique elements or *categories*. As many statistical learning algorithms require numerical representations of features, an encoding step is necessary to transform categorical entries into feature vectors, using for instance *one-hot encoding*. This and other similar strategies work well, in terms of prediction performance and interpretability, in standard statistical analysis when the number of categories is small.
However, non-curated data give rise to string categorical variables with a very high cardinality and redundancy: the string entries share semantic and/or morphological information, and several entries can reflect the same entity. Without any data cleaning or feature engineering step, common encoding methods break down, as they tend to lose information in their vectorial representation. Also, they can create high-dimensional feature vectors, which prevent their usage in large scale settings.
In this work, we study a series of categorical encodings that remove the need for preprocessing steps on high-cardinality string categorical variables. An ideal encoder should be: scalable to many categories; interpretable to end users; and capture the morphological information contained in the string entries.
Experiments on real and simulated data show that the methods we propose improve supervised learning, are adapted to large-scale settings, and, in some cases, create feature vectors that are easily interpretable. Hence, they can be applied in Automated Machine Learning (AutoML) pipelines in the original string entries without any human intervention.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France