PAIS 2022
A. Passerini and T. Schiex (Eds.)

doi:10.3233/FAIA220070

Point of Interest Category Prediction with Under-Specified Hierarchical Labels

Nikolaos LAGOS [a,1], Salah AÏT-MOKHTAR [a], Ioan CALAPODESCU [a] and JinHee LEE [b]

[a] *Naver Labs Europe, Meylan, France*
[b] *Naver Corporation, Seongnam, South Korea*

Abstract. Categories are important elements of databases of Product Listings, for e-commerce platforms, or of Points of Interest (POIs), for location-based services. However, category annotations are often incomplete, which calls for automatic completion. Hierarchical classification has been proposed as a solution to impute missing annotations. We address this task in one of Naver's production databases (POIs), in order to enhance its quality. In real-life applications, like ours, however, it is unrealistic to count on the existence of a perfectly annotated training set, and noisy training labels prevent us from casting the task as a straightforward classification problem. In order to overcome this difficulty, we propose an approach that takes into account the type of noise in the training set. We identified that the main deficiency is that the training labels tend to be under-specified i.e. they point to categories found at higher levels of the hierarchy than the correct ones. This results in a lot of under-represented and a few over-represented categories. We call categories that are over-represented, due to under-specified labels, joker classes. To allow robust learning in the presence of joker classes we propose a simple and effective approach: First, we detect problematic categories, i.e. joker classes, based on the misclassifications of an initial hierarchical classifier. Then we re-train from scratch, introducing a weight to the standard cross-entropy loss function that targets incorrect predictions related to joker classes. Our model has enabled the correction of thousands of POIs in our production database.

Keywords. point of interest, hierarchical classification, label noise

1. Introduction

Categories are important elements of databases of Product Listings, for e-commerce platforms, or of Points of Interest (POIs), for location-based services, such as Google Maps and Naver Maps. In this paper we are particularly interested in POIs, i.e. places that someone may find interesting. POI categories, i.e. tags denoting that a POI is a Falafel Restaurant, Bowling Alley etc., can be used not only to guide humans, but also as input data to several applications such as recommender systems and trip planners. However, in real-life applications, tags are incomplete or imprecise, especially for unpopular or newly-established POIs [1].

[1] Corresponding Author: Nikolaos Lagos, Naver Labs Europe, Meylan, France; E-mail: nikolaos.lagos@naverlabs.com

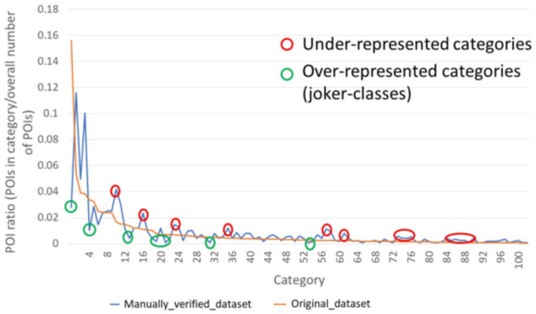


Figure 1. The orange line shows the original distribution of our dataset. The blue line the corrected distribution after our data analysis team performed manual verification, for the top 100 most popular categories, as explained in Section 3. Examples of under and over-represented categories are highlighted. Differences between the two distributions are mainly due to POIs having been annotated with under-specified labels.

ML-based supervised category prediction has been proposed as a solution to impute missing tags [2,3,4,5,6,7,8]. However, as Zhou et al. [6] argue, it is unrealistic to count on the existence of a perfectly annotated training set. This is due to the fact that tags are input either using automatic techniques (e.g. mining user comments), which necessarily comprises errors, and/or by humans who often fail to annotate database items comprehensively, especially when there are thousands of categories to select from. Our data illustrates this as well. In Figure 1, we show how the distribution of the top one hundred most popular categories changes after our data analysis team verified and corrected our original (manually annotated by users) dataset (c.f. Section 3).

In this paper, we propose a method that takes into account the fact that training annotations may be noisy. In particular, we identify that a major source of errors relates to under-specified hierarchical labels i.e. given a label hierarchy, a fully-specified label is one that provides a path from the root node to the most specific correct node (this can be either a leaf or an internal node of the hierarchy). An under-specified label has instead a path that terminates at a category found at higher levels of the hierarchy. This results in a lot of under-represented and a few over-represented categories. We call categories that are over-represented, due to under-specified labels, *joker classes*. For instance, several POIs are tagged with a path terminating close to the top of the hierarchy e.g. "*Restaurant||Korean Food*",[2] while the actual correct path terminates at a lower level e.g."*Restaurant||Korean Food||Seafood||Sliced Raw Fish||Saebyeok Raw fish*". In that case, "*Restaurant||Korean Food*" could be considered a candidate *joker class* and "*Restaurant||Korean Food||Seafood||Sliced Raw Fish||Saebyeok Raw fish*" an under-represented one.

To allow robust learning in the presence of under-specified hierarchical labels we propose a two step approach: 1. First, we automatically detect *joker classes* based on misclassifications of a hierarchical classification model; 2. Then, we introduce a weight to the standard cross-entropy loss function that targets incorrect predictions of joker classes only. More specifically, we penalise misclassifications that correspond to joker classes located higher in the hierarchy than the corresponding training labels, when they share

[2] The symbol || denotes a sub-level in the hierarchy.

similar paths. At the same time, we assign lower cost to misclassifications of categories found lower in the hierarchy than joker class labels.

The main contributions of this work are as follows.

- We propose an approach for robust learning in the presence of under-specified hierarchical labels. To our knowledge this is the first study that identifies this problem in a real-world setting, and illustrates its impact on the quality of corresponding categorisation models.
- We perform experiments on data from our POI database, one of the biggest search and services platforms worldwide. We also report the impact of the deployment of this model on our production database, where the categories of thousands of POIs were corrected.

The rest of the paper is organised as follows. We review related work in Section 2 and the background in Section 3. We define the problem in Section 4 and describe our approach in 5. Experiments are presented in Section 6 and a deployment case study in 7.

2. Related work

2.1. Point-of-Interest classification

Most of the work on Point-of-Interest classification has taken place in the context of Location-Based Social Networks. There are two main approaches to the problem.

The first one requires access to check-in data and uses such data as input to the prediction model [2,3,4]. This includes for instance POI unique identifiers, user unique identifiers, the time and duration of the check-in, the number of check-ins, the latitude/longitude of the user's position, and sometimes users' demographic information (e.g. age range, gender). Based on this information, most of the existing work, attempts to categorise POIs in very coarse-grained categories (e.g. home vs. work, or nightlife/bar vs. restaurant) with the number of categories to predict ranging from 3 to 15. He et al. [5] and Zhou et al. [6], in addition to check-ins, also try to use more fine-grained information about the POIs. In the case of He et al. [5] this includes general tags that may be related to categories but also to other information e.g. ”godzilla”. Zhou et al. [6] use POI name and address tokens or more particularly token embeddings pre-computed on a domain-specific corpus.

Recognising that collecting personal information may be difficult for a large number of POIs, other works are rather based on POI metadata only. Lagos et al. [7] are the first to focus on increasing the POI classifier's coverage by using only the POI name, location, and time of opening attributes. They achieve state-of-art results for a large, multilingual POI database, illustrating that such an approach is feasible. Lie et al. [8] follow that line of work and use only POI names and locations as input to their model. In addition, they propose a voting ensemble of hierarchical classifiers to predict leaf categories. We also use only the name and address of POIs, i.e. no user-related attributes, as inputs. However, contrary to our approach, previous work does not deal with under-specified hierarchical labels. Note that our work can complement techniques based on search queries and user check-ins, but can also be independently used in the case that no such data is available.

2.2. Hierarchical classification

Flat classification approaches ignore the hierarchical relations between categories and treat leaf categories as an independent set of labels. These approaches are easy to implement, but tend to have worse results than hierarchical approaches when labels are organised in a large taxonomy [9]. In contrast, hierarchical classification (HC) systems predict a hierarchically organised path of labels. They are usually divided in local and global approaches [10]: Local approaches learn multiple independent classifiers, each classifier specialised either to each node, parent node or hierarchy level. Global approaches consist of a single model able to map samples to their corresponding category paths as a whole. State-of-art performance has been recently achieved by hybrid approaches combining the local and global paradigms. Wehrmann et al. [11] present a classifier trained with both local and global losses. Giunchiglia and Lukasiewicz [12] propose coherent multilabel classification networks where labels predicted by the local and global classifiers are hierarchically consistent. In this paper, we are also based on a hybrid hierarchical classifier. In addition, we introduce a loss that allows robust learning in the presence of under-specified hierarchical category paths.

3. Background and motivation

POI categorisation, is an important operation that can impact a lot of our location related services. Therefore, enhancing the quality of our category annotations is of high priority.

As a first step to improving category annotations, we implemented and tested the flat POI classifier of Lagos et al. [7].[3]. The classifier encodes POI names and addresses using 1-gram character-based LSTMs and feeds the corresponding embeddings to a simple feedforward neural network. [4] In addition to the standard micro and macro prediction quality metrics [15] calculated on the development dataset (c.f. Section 6.1), it was extremely important to qualify the errors made by the system before being able to deploy it. Most importantly, knowing that we have a very long tail in our data distribution, we had to understand the behaviour of the system on the corresponding POIs. As illustrated in Figure 2, while the system was doing well on category paths in the middle of the range, and comparatively well for paths with few POIs (the performance was deteriorating when only one POI was related to a specific category path), the performance was lower than expected on paths that were heavily populated.

To qualify better the errors we extracted a set of 1000 misclassifications for further analysis. The sample was representative of the prediction probabilities one could find in the misclassifications of the development set i.e. if 15% of the misclassifications on the development set had a probability of over 0.9, then we kept the same ratio in the set we extracted. Details of the analysis are presented in Table 1.

As shown in Table 1, misclassifications, i.e. disagreements between the labels predicted by the ML model and the ones found in the development dataset (which included

[3] This classifier is very competitive in our setting, even when compared to other, attention-based, flat-classification alternatives. Details can be found in our technical report [13]

[4] It is important to note here, that a word-based representation didn't perform better. We assume this is partly due to the fact the we deal with Korean (NFC encoding), where one character is actually similar to a syllable, rather than a single letter, in Indo-European languages [14].

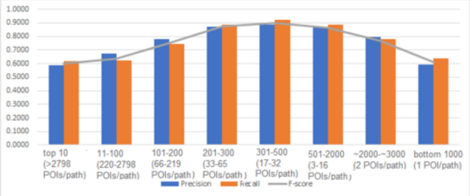


Figure 2. Performance of the initial flat hierarchical classifier in relation to the number of POIs attributed to category paths.

Table 1. Verification of disagreements between the silver test dataset and the prediction generated by our flat classifier. The model is able to correct the human annotations at high probability thresholds, or recommend correct alternative tags. The resulting verified dataset is considered as the gold standard in the rest of the paper.

Probability	Correct(%)	Acceptable(%)
>0.9	66.63	7.62
0.7-0.9	∼32	7-12
0.4-0.7	13-15	31-40
<0.4	<3	<5

existing, manually entered, data), were largely due to incorrect labels in the development dataset. Errors in the original label annotations were the cause of almost up to two thirds of the misclassifications when the probability given by the ML model was over 0.9, and almost of one third when the value was between 0.7 and 0.9. Ontological and multi-labeling issues (i.e. two different categories being correct for the same POI), were also discovered. For instance, semantically similar categories, such as ||*Cafe, Dessert*||*Cafe*[5] and ||*Cafe, Dessert*, accounted up to one third of the model's misclassifications when the output probability was between 0.4 and 0.7 (*Acceptable* column in Table 1).

Based on the above analysis we found that:

- The distribution of labels in the verified, corrected dataset is different from the one in our existing, manually entered, data. More specifically, several POIs initially attributed to over-represented classes were re-attributed to more specific category paths , and most importantly to long-tail classes [6]. In our experiments we use the verified, corrected dataset as our gold standard.
- Minimising the number of under-specified labels is particularly important for improving data pertinence e.g. tagging a POI as ||*Korean Food*||*Seafood*||*Sliced Raw Fish*||*Sashimi* instead of ||*Korean Food* is much more informative.
- Setting an appropriate threshold for the ML output probabilities is paramount. In our case, we set it at 0.4.
- The ML model can be useful not only for imputing labels for new POIs but also for curating existing annotations, in a principled manner.

[5] || denotes a sub-level in the hierarchy. We omit the root category *Restaurant* for clarity.

[6] Details can found in our technical report [13].

4. Problem definition

In our setting, a POI p is represented as $p = \{\boldsymbol{x}, y\} = \{x^{(1)}, x^{(2)}, y\}$ where $\boldsymbol{x}$ is a vector, representing POI's name, $x^{(1)}$, and address, $x^{(2)}$, as well as a label y, representing a hierarchical category path.

We assume a tree structured hierarchy of categories $T = (C, E)$ where $C = \{c_0^0, .., c_n^k\}$ is the set of n pre-defined categories with a maximum depth of k, such that $E = \{(c_\ell^h, c_j^{h+1}) \in C | c_\ell^h \prec c_j^{h+1}, h \leq k\}$, where h is an index indicating the level of the hierarchy (i.e. hierarchy depth) and $\prec$ denotes the sub-category-of relation. For instance, if we have the root-to-leaf path of categories *"Restaurant||Korean Food||Seafood||Sliced Raw Fish||Sashimi"*, as shown in Figure 3, and c_ℓ^h is the path *"Restaurant||Korean Food"* then c_j^{h+1} would be the path *"Restaurant||Korean Food||Seafood"*.

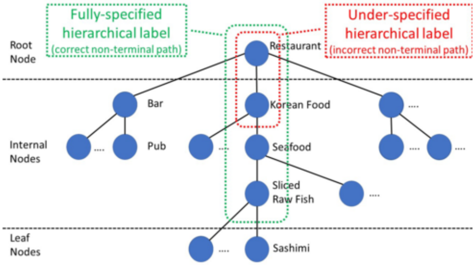


Figure 3. Example of under-specified and fully-specified hierarchical labels. Notice that a fully-specified hierarchical label can correspond to a correct non-terminal path of categories i.e. a path that terminates at an internal node. Our objective is to design a classifier robust to under-specified hierarchical labels.

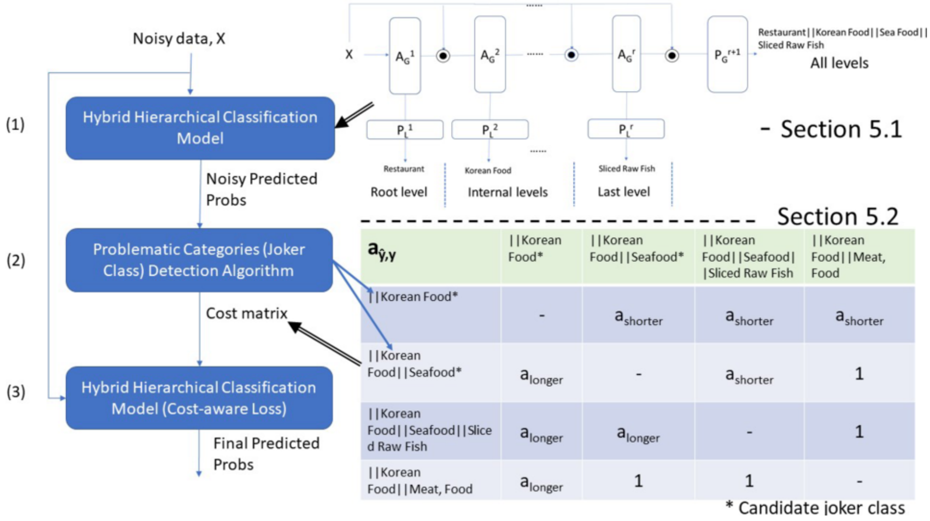
Given T, y should ideally represent a fully-specified path of categories $t = (c^0, c^1, ..., c^m)$.[7] Note that in our setting, we may also have correct non-terminal paths i.e. $m < k$, which means that a fully-specified correct path does not have to include categories up to the leaf nodes of the hierarchy, but it can instead terminate at an internal node. Back to our example, the path *"Restaurant||Korean Food||Seafood||Sliced Raw Fish"* could be correct if the corresponding POI served several different types of sliced raw fish. In addition, and most importantly, in our real-world case we find that observed paths $t' = (c^0, ...c^z)$, in the training data, may be under-specified i.e. $z < m$, and thus incorrect. For instance, t' could be *"Restaurant||Korean Food"*. Our objective is to design a classifier robust to the above data characteristics.

5. Our Approach

To allow robust learning in the presence of annotations with under-specified hierarchical labels, we propose the approach illustrated in Figure 4: (1) We develop a hybrid hierarchical classifier that combines one global and potentially several local classifiers (c.f. Section 5.1) using standard categorical cross-entropy losses; (2) We automatically detect problematic categories i.e. candidate **joker classes**, (c.f. Section 5.2.2) based on the misclassifications of the classifier of step (1); and (3) We introduce a cost-based weight to

[7] We omit subscripts for clarity.

the global classifier's loss and re-train the model from scratch. The weight specifically penalises misclassifications having shorter category paths than the ones found in the corresponding human annotations, while accordingly it assigns lower cost to misclassifications having longer category paths.



$a_{\hat{y},y}$	||Korean Food*	||Korean Food||Seafood*	||Korean Food||Seafood||Sliced Raw Fish	||Korean Food||Meat, Food
||Korean Food*	-	$a_{shorter}$	$a_{shorter}$	$a_{shorter}$
||Korean Food||Seafood*	a_{longer}	-	$a_{shorter}$	1
||Korean Food||Seafood||Sliced Raw Fish	a_{longer}	a_{longer}	-	1
||Korean Food||Meat, Food	a_{longer}	1	1	-

* Candidate joker class

Figure 4. Our Approach. (1). We train a hybrid hierarchical classifier that combines one global (represented with the suffix $_G$ in the Figure) and potentially several local classifiers (represented with the suffix $_L$ in the Figure) (c.f. Section 5.1); (2). We automatically detect problematic categories i.e. candidate **joker classes**, (c.f. Section 5.2.2) based on the misclassifications of the classifier trained at step (1). For instance, *"||Korean Food"* and *"||Korean Food||Seafood"* are such candidates as shown in the Figure; (3). We introduce a cost to the global classifier's loss, which accounts for misclassifications involving joker classes, and repeat step (1) (i.e. re-train the model from scratch with the same configuration as in step (1)), as described in Sections 5.2.1 and 5.2.2. For instance, when *"||Korean Food||Seafood"* is the predicted label, and the observed label is *"||Korean Food"*, then the cost is a_{longer}. On the other hand, if the observed label represents a longer path, e.g. *"||Korean Food||Seafood||Sliced Raw Fish"* than the predicted one, then the cost is $a_{shorter}$.

5.1. Hybrid hierarchical classification model

Wehrmann et al. [11] have shown that a hierarchical classifier that performs both local and global optimisation, has significant advantages over adopting one of the two approaches only. Following [11], we implement a multiple-output deep neural network, as illustrated in Figure 4. In the Figure, each A_G^h represents intermediate (hidden) layers, where h index the index of the hierarchical level. For instance, $h = 1$ corresponds to the root node (there may be several root nodes in the data). Each P_x^h represents an output layer, where $x = L$ indicates the output of a local classifier and $x = G$ the output of the global classifier. As shown in the Figure, the architecture has one local output per hierarchical level, with a corresponding local loss function $\mathscr{L}_L^h$, and one global output for the full category path, with a global loss $\mathscr{L}_G$. The input of the first local classifier, indicated as X, corresponds to the same 1-gram character-based LSTM embeddings given to the flat classifier of Lagos et al. [7], as described in Section 3. Each local classifier thereafter has as input the concatenation of the initial inputs and an intermediate embedding representing the feature space of the previous local classifiers i.e. a dense layer, before the output layer, of the previous local classifier. Dense layers are activated with a non-linear function (i.e. ReLU). The output layer of the global classifier has as input the concatenation of the initial inputs and the embedding given also to the last local classifier A_G^r, which as Wehrmann et al. [11] highlights is the cumulative information of the feature space of all local classifiers, concatenated with the initial inputs. The final loss is the sum of the global output loss $\mathscr{L}_G$ and all local output losses $\mathscr{L} = \mathscr{L}_G + \sum_{h=1}^{r} \mathscr{L}_L^h$, where $r \leq k$. As we want the classes to be mutually exclusive for each level, we use the standard categorical cross-entropy loss for each $\mathscr{L}_L^h$ and for $\mathscr{L}_G$.

Differently to Wehrmann et al. [11], we also want to account for non-terminal paths i.e. observed paths that do not terminate at a leaf node but at an internal one. To achieve that, we (i) use a special category token to denote when the end of a non-terminal path has been reached (ii) we allow $r < k$, thus effectively allowing the implementation of different networks that incrementally cover more levels of the hierarchy, until an optimal depth is found.

5.2. *Cost-based approach for incomplete category paths*

5.2.1. *Joker class-agnostic cost*

To account for under-specified category paths, we propose to penalise more misclassifications with shorter paths than the ones observed in the training data, when that shorter path is shared by both the prediction and the observed label, than misclassifications with longer paths. In addition, we want the latter case to be penalised less than the rest of the errors i.e. when the prediction and observed label do not share, at least in part, a common path. For instance, assume that y represents the path $t' =$ "*Restaurant*||*Korean Food*||*Seafood*". If $\hat{y}$ denotes the prediction with shorter path $\hat{t} =$ "*Restaurant*||*Korean Food*", then we want this prediction to be penalised more than if it represented the one with longer path "*Restaurant*||*Korean Food*||*Seafood*||*Sliced Raw Fish*". Specifically, let $a_{\hat{y}_i,y_i}$ denote the cost associated with assigning the label $\hat{y}$ to the sample i that has an observed label y. If we denote $a_{shorter}$ the cost of predicting a shorter path than the observed one and a_{longer} the cost of predicting a longer path, then $a_{longer} < a_{shorter}$. Both a_{longer} and $a_{shorter}$ are set empirically[8]. Accordingly, we change $\mathscr{L}_G$ from the standard categorical cross-entropy to the following weighted-by-sample loss.

$$\mathscr{L}'_G = \frac{1}{N} \sum_{i=1}^{N} a_{\hat{y}_i,y_i} \mathscr{L}_{G,i} \tag{1}$$

where $\mathscr{L}_{G,i}$ is the standard global categorical cross-entropy loss for sample i and

$$a_{\hat{y}_i,y_i} = \begin{cases} a_{shorter}, & \text{if } \hat{t}_i.prefix\_path\_of(t'_i) \\ a_{longer}, & \text{if } t'_i.prefix\_path\_of(\hat{t}_i) \\ 1, & \text{otherwise.} \end{cases}$$

[8] Based on grid search: (i) a_{longer}, range 0.1-1.0 with a step of 0.1 (ii) $a_{shorter}$, range 1.0-10.0 with a step of 0.1 between 1.0-2.0, and then with a step of 0.5 for the range 2.0-10.0. Detailed ablation study is included in our technical report [13].

Above, $\hat{t}_i$ is the path corresponding to the $\hat{y}_i$ prediction and t'_i is the observed path corresponding to y_i. The *prefix_path_of* function indicates a ”strict” prefix i.e. the two paths cannot be identical.

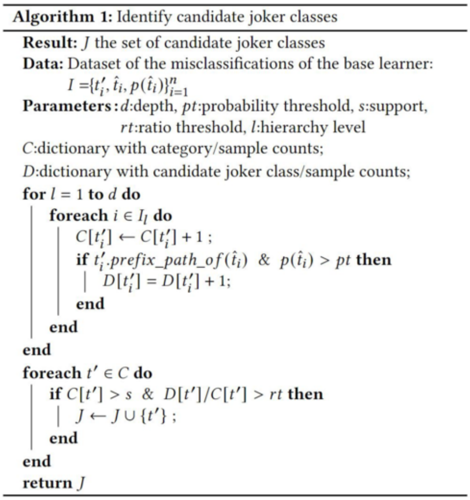
5.2.2. Joker class-specific cost

The global loss defined in the previous section applies to all category paths. However, only a small set of unique paths correspond to the majority of real incorrect misclassifications, the ones we referred to in Section 1 as *joker classes*. By applying the cost in a joker class-agnostic manner, we over-punish training samples related to non-joker classes (i.e. intuitively, we make the model less confident for some correct annotations).

To tackle this issue, we propose to automatically identify candidate joker classes based on the misclassifications of the initial hierarchical classification model and apply the a_{longer} and $a_{shorter}$ costs only to samples that have labels corresponding to these classes. More specifically, we rely on the following assumptions:

- The first one is related to a widely-used assumption in weakly-supervised learning [16] : Although the hierarchical classification model, i.e. our base learner, is not optimal, it is still able to predict the correct category for the majority of the samples for which the model is most confident. We consider as an indication of confidence the probability the model assigns to a prediction.
- The second assumption is related to an observation: As found in a preliminary qualitative analysis of the development set, the majority of the misclassifications for which our base learner is very confident (i.e. probability >0.9) is related to predictions that have longer paths than the ones found in the corresponding manual annotations. So, finding frequent paths related to misclassifications with a high prediction probability, can be an indication that they correspond to *joker classes*.

Based on the above, finding candidate *joker classes* amounts to identifying category paths that are frequently misclassified by the model with a high certainty, as shown in Algorithm 1. In Algorithm 1, we also define the minimum support s, i.e. the minimum number of samples that should be related to a category, and the depth d to which the



```
Algorithm 1: Identify candidate joker classes
Result: $J$ the set of candidate joker classes
Data: Dataset of the misclassifications of the base learner:
        $I = \{t'_i, \hat{t}_i, p(\hat{t}_i)\}_{i=1}^{n}$
Parameters: $d$:depth, $pt$:probability threshold, $s$:support,
            $rt$:ratio threshold, $l$:hierarchy level
$C$:dictionary with category/sample counts;
$D$:dictionary with candidate joker class/sample counts;
for $l = 1$ to $d$ do
    foreach $i \in I_l$ do
        $C[t'_i] \leftarrow C[t'_i] + 1$ ;
        if $t'_i.prefix\_path\_of(\hat{t}_i)$ & $p(\hat{t}_i) > pt$ then
            $D[t'_i] = D[t'_i] + 1$;
        end
    end
end
foreach $t' \in C$ do
    if $C[t'] > s$ & $D[t']/C[t'] > rt$ then
        $J \leftarrow J \cup \{t'\}$ ;
    end
end
return $J$
```

search for joker classes should stop. The latter is related to the nature of our problem: as most joker classes tend to be located at higher levels of the hierarchy by definition, the benefit of the algorithm is becoming potentially less important as we are moving lower in the hierarchy, while the real misclassification error rate tends to increase. In our experiments we set d to 3, i.e. we consider the top half of the hierarchy, and s is set empirically to 100. We set $pt > 0.9$ and rt to the median of the remaining categories, resulting in 22 candidate joker classes.

6. Experimental setting

The focus of the experiments was twofold (i). evaluate the capability of the model to predict POI categories, and (ii). understand the impact of under-specified labels.

6.1. Data

We run our experiments on a large dataset including 828K POIs and 4093 unique category paths (we count only paths that appear as the label of at least one POI). Each POI is labelled with exactly one path. The maximum depth of the categorisation hierarchy is 5. The hierarchy is very fine-grained. For instance, there are 70 sub-categories of the *Pizza* category located at the third level in the hierarchy. To our knowledge this is the first work dealing with such a rich hierarchy in the POI classification domain.[9]

The dataset is heavily imbalanced in terms of the number of POI instances attributed to each category [10]. For instance, the top ten categories have more than 40% of the POIs attributed to them, while 461 categories have fewer than 5 POIs, resulting in a very long queue of sparsely represented categories.

To generate training and test data, we used stratified sampling. Consequently, we allocated the 828K POIs proportionally into 70% for training, 20% for development, and we kept 10% for future testing purposes. The gold standard includes 1000 POIs that we carefully verified. It is important to highlight that the **gold standard contains only misclassifications of the preliminary flat classifier as per the silver standard, thus consists exclusively of samples that are very difficult to categorise.** Figure 1, in Section 1, shows the distribution of the gold standard when compared to the silver one.

6.2. Models and implementation details

The models we evaluate are described below:

Base [7]: We use the flat classifier of Lagos et al. [7] as our baseline. In detail, we have implemented an architecture with one hidden dense layer, followed by a dropout layer, and a softmax output layer. We use the Rectified Linear Unit as the activation function of the hidden layer. The dropout rate is set to 0.3. The loss we use is categorical crossentropy. We have set an early stopping criterion for the training based on a predefined threshold that takes into account the delta of the loss between two consecutive epochs. We set the maximum number of epochs to 50. We consider both POI attributes

[9] The actual characteristics of our dataset are closer to previous work in item classification for online product listings [17].

[10] Details can be found in our technical report [13].

as sequential features with a length of 50. For the LSTM layer we set the dimensions of the embedding layer vector space to 128 and the number of the LSTM hidden units to 128. The LSTM has recurrent droupout rate of 0.3. The rest of the models have the same hyper-parameter values.

Hcls: Hcls stands for the hybrid hierarchical model described in Section 5.1. In Table 2, we report only the model that performs the best (the one that takes into account only the second level of the hierarchy)[11]. After each dense layer that corresponds to a hierarchical categorisation level (c.f. Section 5.1), we insert a dropout layer (dropout rate of 0.3). The rest of the implementation details are the same as for the baseline.

Focal,**cb**: Given the fact that a lot of POIs are re-attributed in our gold dataset to long-tail categories, we have also implemented state-of-art approaches that counter the effect of a skewed data distribution by adjusting the weights of the samples from the small classes in the loss function . In that context, focal-loss [18], as well as its combination with class-balanced loss have shown the most promising results recently [19]. Focal loss adds a modulating parameter γ to the cross-entropy loss to allow focusing on long tail samples. Class-balanced loss offers an alternative to using inverse class frequency, by introducing the concept of the class effective number via the hyperparameter β, which is used to calculate the weight of each class in the loss term [19]. We replace the categorical cross-entropy loss of the global output with these losses in our setting. We set the modulating parameter γ of focal loss at 1.0 and the β parameter of the class-balanced loss at 0.2, after performing a grid search with step size of 0.1. We refer the reader to [18] and [19] for more details on these methods.

C$\mathscr{L}$: C$\mathscr{L}$ stands for the cost-based loss introduced in Section 5.2.1, which is agnostic to the classes that the samples belong. The cost for misclassifications where the predicted labels have longer paths than the observed ones, is set to 0.5, while the cost for misclassifications with shorter paths is set to 1.4.

CJ$\mathscr{L}$: CJ$\mathscr{L}$ corresponds to the joker-class specific loss introduced in Section 5.2.2. The suffixes indicate the cost values used. For instance, CJ$\mathscr{L}_{1.4,0.5}$, corresponds to the same cost values as the ones used for the C-$\mathscr{L}$ model, while CJ$\mathscr{L}_{5.0,0.5}$, means that the cost given to misclassifications with shorter paths is 5.0.

All experiments were run on a single GPU instance (1 GPU with 16GB VRAM, 4 CPUs, with 256GB RAM). Training was performed with a batch size of 128. We used the Adam optimiser with the default parameters. We use the standard macro and micro metrics for the evaluation calculated using the scikit-learn package [15].

6.3. Results

Our models achieve the best results, as shown in Table 2. We achieve improvements up to 4.89% in micro-F1 and 3.98% in macro-F1 on the gold standard. Please note that the absolute scores may seem quite low, however, the gold standard consists exclusively of a subset of examples that the preliminary flat model fails to classify correctly as per the silver dataset, thus being very difficult to categorise (c.f. Section 3).

The Hcls+CJ$\mathscr{L}$ models performs better than the class-agnostic model, *Hcls+C*$\mathscr{L}$, especially in terms of micro-F1. The difference is less important in macro-F1, as *Hcls+C*$\mathscr{L}$ applies costs to all misclassifications, implicitly pushing the model to predict long-tail categories in a stronger manner than in the case of the *Hcls+CJ*$\mathscr{L}$ models.

[11] An ablation study is included in our technical report [13].

Table 2. Average performance (%) over 5 runs on the silver and gold standards. Best results per dataset are in **bold**. Standard deviation is also reported. Hcls+CJ$\mathscr{L}_{1.4,0.5}$ performs well on both the silver and gold standards. Hcls+CJ$\mathscr{L}_{5.0,0.5}$ has the best overall performance on the gold standard.

Dataset	Silver standard			Gold standard		
Model	Metric					
	Micro-prec.	Micro-rec.	Micro-F1	Micro-prec.	Micro-rec.	Micro-F1
Base [7]	70.41±0.27	63.78±0.26	66.93±0.2	46.57±2.02	32.62±0.47	38.36±0.88
Hcls	72.85±0.26	63.90±0.26	68.08±0.05	50.32±1.14	31.99±0.46	39.11±0.58
+Focal	72.64±0.18	63.59±0.23	67.81±0.06	51.39±0.85	32.41±0.45	39.75±0.51
+Focal-cb	72.96±0.26	63.30±0.31	67.79±0.07	50.99±0.73	32.00±0.84	39.32±0.79
Ours						
+C$\mathscr{L}$	73.27±0.41	62.90±0.59	67.69±0.22	52.67±0.51	31.94±1.00	39.76±0.78
+CJ$\mathscr{L}_{1.4,0.5}$	73.34±0.4	63.37±0.27	67.99±0.17	53.60±1.17	**33.51±0.26**	41.23±0.4
+CJ$\mathscr{L}_{1.4,-}$	**73.70±0.17**	62.97±0.18	67.91±0.05	53.67±1.64	32.31±0.34	40.34±0.70
+CJ$\mathscr{L}_{-,0.5}$	72.84±0.30	**63.93±0.28**	**68.09±0.03**	51.41±1.71	33.22±0.91	40.36±1.01
+CJ$\mathscr{L}_{5.0,0.5}$	73.58±0.57	59.09±0.37	65.54±0.43	**62.62±0.99**	33.04±0.77	**43.25±0.56**
	Macro-prec.	Macro-rec.	Macro-F1	Macro-prec.	Macro-rec.	Macro-F1
Base [7]	79.18±1.00	80.38±0.86	78.72±0.91	47.75±1.6	37.80±1.39	39.82±1.33
Hcls	80.29±0.18	82.23±0.51	80.08±0.32	49.66±1.69	38.38±1.10	40.92±1.33
+Focal	79.77±1.12	81.11±0.61	79.19±0.62	49.10±1.63	38.34±0.72	40.86±0.59
+Focal-cb	80.23±0.69	82.01±0.58	79.83±0.39	49.34±0.83	38.40±0.86	40.77±0.78
Ours						
+C$\mathscr{L}$	80.50±0.36	82.89±0.18	80.51±0.29	50.60±1.5	40.87±1.4	43.08±1.04
+CJ$\mathscr{L}_{1.4,0.5}$	80.72±0.2	83.46±0.29	80.87±0.22	51.04±0.92	40.54±0.51	42.98±0.41
+CJ$\mathscr{L}_{1.4,-}$	80.65±0.13	83.10±0.22	80.67±0.09	**52.12±0.77**	41.48±0.82	**43.93±0.74**
+CJ$\mathscr{L}_{-,0.5}$	**80.76±0.14**	**83.50±0.43**	**80.90±0.14**	49.94±0.67	40.37±0.21	42.57±0.36
+CJ$\mathscr{L}_{5.0,0.5}$	80.34±0.48	83.37±0.39	80.55±0.42	51.04±1.66	**41.50±1.45**	43.80±1.51

Because of that, more POIs from head categories[12], are wrongly misclassified, resulting in the drop in micro-F1. To some extent, the additional POIs attributed to long-tail categories[13], smooth out this difference in macro-F1.

The two costs related to shorter and longer paths have different impacts. As shown in Table 2, keeping only the cost related to shorter paths, i.e. $CJ\mathscr{L}_{1.4,-}$, gives better micro-precision scores than when the cost for longer paths is used, i.e. $CJ\mathscr{L}_{-,0.5}$. On the other hand, micro-recall mainly benefits from the cost given to longer paths. On the macro-scores, the latter cost does very well on the silver standard, while the former on the gold standard. The combination of both costs i.e. $CJ\mathscr{L}_{1.4,0.5}$, does not always give the best score, but achieves a balanced performance, in terms of F1.

On the silver standard, the cost-related models achieve better results in terms of macro-F1, gaining up to 2.18 points compared to the baseline and 0.82 points to the

[12] Most heavily populated categories.

[13] Less heavily populated categories.

initial hierarchical model, *Hcls*. This is due to long-tail POIs being predicted more often. On the other hand, the *Hcls* model has comparable (or even slightly better) micro-F1. This is not surprising, considering that the silver standard shares the same issue of *joker classes* with the training data. The results on the gold standard, where the cost-related models have significantly better scores than *Hcls*, is also a strong indication of that.

Surprisingly, the *Hcls+focal-cb* model does not have singificantly better scores when compared to the initial *Hcls* model on the gold standard. More work is required to understand the underlying reasons. We note that to our knowledge this is the first work combining hierarchical classifiers with data imbalance losses. A detailed ablation study can be found in our technical report [13].

7. Deployment case study

We developed the classification system and model for the dataset described in this paper in September 2020. We have released the model for the full POI database in November 2020. Among other things, corresponding predictions were used to semi-automatically correct existing annotations in the production database. For instance, on the dataset used in this paper alone, $>$11.4K POIs were corrected in just a few days, as we focused on misclassifications with probability greater than 0.9 ($\approx$17K predictions). Corrected POIs are currently available online via Naver Maps. Note that data curation is resource intensive, so having a tool that allows to organise it in a principled manner is very useful.

8. Conclusions

In this paper, we presented our approach to POI category prediction in Naver. In this setting, category prediction is a hierarchical classification problem, as categories are organized in a detailed tree structure. However, in real-life applications, like ours, it is unrealistic to count on the existence of a perfectly annotated training set. This prevented us from casting the task as a straightforward classification problem. In order to overcome this difficulty, we identified that the main deficiency of the training set is that the training labels tend to be under-specified i.e. they point to categories found at higher levels of the hierarchy than the correct ones. This resulted in a lot of under-represented and a few over-represented categories. To allow robust learning in that context we proposed the following approach: First, we detected problematic categories, i.e. over-represented categories, based on the misclassifications of an initial hierarchical classifier. Then we re-trained from scratch, introducing a weight to the standard cross-entropy loss function that specifically targeted incorrect predictions of these categories only. We find that on our gold standard we achieve improvements up to 4.89% in micro-F1 and 3.98% in macro-F1. Accordingly, our system has enabled the correction of thousands of POIs in our production database, showing that the resulting classifier may be used not only to impute categories to new POIs, but also to curate and correct manually added ones.

References

[1] Zhou J, Gou S, Hu R, Zhang D, Xu J, Jiang A, et al. A Collaborative Learning Framework to Tag Refinement for Points of Interest. In: Proc. 25th ACM SIGKDD. KDD '19; 2019. p. 1752–1761.

[2] Ye M, Shou D, Lee WC, Yin P, Janowicz K. On the Semantic Annotation of Places in Location-based Social Networks. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '11. New York, NY, USA: ACM; 2011. p. 520-8. Available from: http://doi.acm.org/10.1145/2020408.2020491.

[3] Wang Y, Qin Z, Pang J, Zhang Y, Xin J. Semantic Annotation for Places in LBSN Through Graph Embedding. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. CIKM '17. New York, NY, USA: ACM; 2017. p. 2343-6.

[4] Krumm J, Rouhana D. Placer: Semantic Place Labels from Diary Data. In: Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing. UbiComp '13. New York, NY, USA: ACM; 2013. p. 163-72. Available from: http://doi.acm.org/10.1145/2493432.2493504.

[5] He T, Yin H, Chen Z, Zhou X, Sadiq S, Luo B. A Spatial-Temporal Topic Model for the Semantic Annotation of POIs in LBSNs. ACM Trans Intell Syst Technol. 2016 Jul;8(1):12:1-12:24. Available from: http://doi.acm.org/10.1145/2905373.

[6] Zhou J, Gou S, Hu R, Zhang D, Xu J, Wu X, et al. A Collaborative Learning Framework to Tag Refinement for Points of Interest. 2019. Accepted at KDD 2019.

[7] Lagos N, Ait-Mokhtar S, Calapodescu I. Point-Of-Interest Semantic Tag Completion in a Global Crowdsourced Search-and-Discovery Database. In: Giacomo GD, Catalá A, Dilkina B, Milano M, Barro S, Bugarín A, et al., editors. ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020). vol. 325 of Frontiers in Artificial Intelligence and Applications. IOS Press; 2020. p. 2993-3000. Available from: https://doi.org/10.3233/FAIA200474.

[8] Liu S, Yu J, Li J, Hou L. Geographical Information Enhanced POI Hierarchical Classification. In: Wang G, Lin X, Hendler J, Song W, Xu Z, Liu G, editors. Web Information Systems and Applications. Cham: Springer International Publishing; 2020. p. 108-19.

[9] Babbar R, Partalas I, Gaussier E, Amini MR. On Flat versus Hierarchical Classification in Large-Scale Taxonomies. In: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2. NIPS'13. Red Hook, NY, USA: Curran Associates Inc.; 2013. p. 1824–1832.

[10] Masera L, Blanzieri E. AWX: An Integrated Approach to Hierarchical-Multilabel Classification. In: Berlingerio M, Bonchi F, Gärtner T, Hurley N, Ifrim G, editors. Machine Learning and Knowledge Discovery in Databases. Springer International Publishing; 2019. p. 322-36.

[11] Wehrmann J, Cerri R, Barros R. Hierarchical Multi-Label Classification Networks. In: Dy J, Krause A, editors. Proceedings of the 35th International Conference on Machine Learning. vol. 80 of Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR; 2018. p. 5075-84. Available from: http://proceedings.mlr.press/v80/wehrmann18a.html.

[12] Giunchiglia E, Lukasiewicz T. Coherent Hierarchical Multi-Label Classification Networks. ArXiv. 2020;abs/2010.10151.

[13] Lagos N, Ait-Mokhtar S, Calapodescu I, Lee J. Technical Report: Point of Interest Category Prediction with Under-Specified Labels; 2022. https://europe.naverlabs.com/research/systemic-ai/t-of-interest-category-prediction-with-under-specified-hierarchical-labels/.

[14] Park S, Byun J, Baek S, Cho Y, Oh A. Subword-level Word Vector Representations for Korean. In: Proceedings of the 56th Annual Meeting of the ACL; 2018. p. 2429-38.

[15] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011;12:2825-30.

[16] Yarowsky D. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In: 33rd Annual Meeting of the Association for Computational Linguistics. Cambridge, Massachusetts, USA: ACL; 1995. p. 189-96. Available from: https://www.aclweb.org/anthology/P95-1026.

[17] Ha JW, Pyo H, Kim J. Large-Scale Item Categorization in e-Commerce Using Multiple Recurrent Neural Networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, NY, USA: Association for Computing Machinery; 2016. p. 107-15. Available from: https://doi.org/10.1145/2939672.2939678.

[18] Lin T, Goyal P, Girshick R, He K, Dollar P. Focal Loss for Dense Object Detection. In: 2017 IEEE International Conference on Computer Vision (ICCV); 2017. p. 2999-3007.

[19] Cui Y, Jia M, Lin T, Song Y, Belongie S. Class-Balanced Loss Based on Effective Number of Samples. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2019. p. 9260-9.