Point of Interest Category Prediction with Under-Specified Hierarchical Labels

Nikolaos Lagos
nikolaos.lagos@naverlabs.com
Naver Labs Europe
Meylan, France

Salah Aït-Mokhtar
salah.ait-mokhtar@naverlabs.com
Naver Labs Europe
Meylan, France

Ioan Calapodescu
ioan.calapodescu@naverlabs.com
Naver Labs Europe
Meylan, France

JinHee Lee
iam.jinhee.lee@navercorp.com
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, robust learning, label noise, underspecified labels

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 [28].

ML-based supervised category prediction has been proposed as a solution to impute missing tags [10, 12, 14, 17, 24, 27, 28]. However, as Zhou et al. [28] argues, 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).

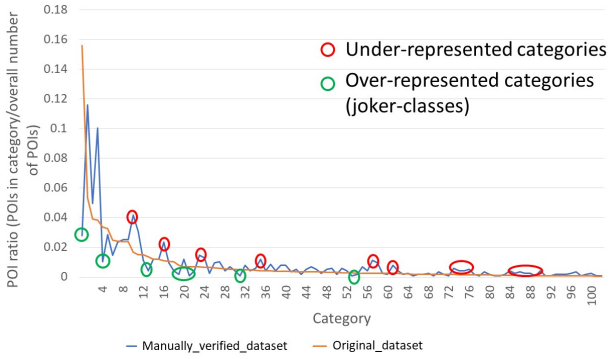


Figure 1: The orange line shows the original distribution of our dataset. The blue line the corrected distribution after our data analysis team performed manual verification, for the top 100 most popular categories, as explained in Section 3. Examples of under and over-represented categories are highlighted.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

In this paper, we propose a method that takes into account the fact that training annotations may be noisy. Most particularly, 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*",[1] 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 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 extensive 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. We define the problem in Section 4 and describe our approach in Section 5. Experiments are presented in Section 6 and a deployment case study in 7. Section 8 includes the conclusions.

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 [12, 24, 27]. 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 no. of categories to predict ranging from 3 to 15. He et al. [10] and Zhou et al. [28], in addition to check-ins, also try to use more fine-grained information about the POIs. In the case of He et al. [10] this includes general tags that may be related to categories but also to other information e.g. "godzilla". Zhou et al. [28] 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. [14] 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. [17] 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 other 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 [2]. In contrast, hierarchical classification (HC) systems predict a hierarchically organised path of labels [1]. They are usually divided in local and global approaches [23]: Local approaches learn multiple independent classifiers, each classifier specialised either to each node [7], parent node [13] or hierarchy level [4]. Global approaches consist of a single model able to map samples to their corresponding category paths as a whole [18]. State-of-art performance has been recently achieved by hybrid approaches combining the local and global paradigms. Wehrmann et al. [25] present a classifier trained with both local and global losses. Giunchiglia and Lukasiewicz [8] 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 following the work of Wehrmann et al. [25]. 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. [14].[2]. The classifier encodes POI names and addresses using 1-gram character-based LSTMs and feeds the corresponding embeddings to a simple

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

[2]As shown in Appendix A, this classifier is very competitive in our setting, even when compared to other, attention-based, flat-classification alternatives.

feedorward neural network.[3,4] In addition to the standard micro and macro prediction quality metrics [21] calculated on the development dataset (c.f. Section 6.1), shown for reference in Table 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. Figure 2 illustrates the performance of the system in relation to the number of POIs attributed to corresponding category paths. While the system was doing very well on category paths in the middle of the range, and comparatively well for paths with very 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.

Table 1: Performance of the preliminary flat classifier with no prediction probability threshold applied. All micro-scores are the same, as only the top ranked prediction for each POI is selected, resulting in the same number of predicted and true labels.

	Precision	Recall	F1
Micro_preliminary	65.24		
Macro_preliminary	76.02	77.60	75.59

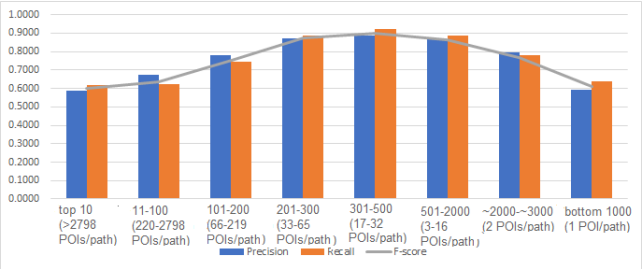


Figure 2: Performance of the initial flat hierarchical classifier in relation to the number of POIs attributed to category paths.

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 2.

As shown in Table 2, misclassifications, i.e. disagreements between the labels predicted by the ML model and the ones found

[3]We consider each category path found in the training data, as a distinct, independent label. Thus, although we predict the full path, rather than just single leaf categories, we consider the classifier as a flat one, rather than as a global hierarchical one.

[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 [20].

Table 2: Verification of disagreements between the silver test dataset and the prediction generated by our initial flat classifier. The model is able to correct the human annotations at high probability threshold levels, 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

in the development dataset (which included 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. In addition, ontological and multi-labeling issues (i.e. two different categories being correct for the same POI), were 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 (we denoted such cases in Table 2 as *acceptable*).

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, as shown in Table 3, and most importantly to a large number of long-tail classes. 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. We, thus, gained more than 1.4 points in micro-f1, compared to using the standard value of 0.5, as shown in Appendix A.[6]
- The ML-based categorisation model can be useful not only for imputing labels for new POIs but also for curating existing annotations, in a principled manner.

4 PROBLEM DEFINITION

In our setting, a POI p is represented as $p = \{x, y\} = \{x^{(1)}, x^{(2)}, y\}$ where 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.

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

[6]*Lagos et al. [15] model.*

[7]All re-attributed POIs are given the alternative tag *||Cafe, Dessert||Cafe.*

[8]The majority of alternative tags have the value *||Korean Snack||Gimbap.*

Table 3: Percentage of POIs attributed to categories lower in the hierarchy after verification, for some heavily populated categories. The symbol "||" indicates a sub-level in the hierarchy. We use * to denote the percentage of tags that were re-attributed to a different category because of ontological or multi-labeling issues.

Category	Re-attributed POIs(%)
||Korean Food	80.29 (1.46*)
||Cafe, Dessert[7]	59.3*
||Cafe, Dessert||Cafe	0.0
||Korean Food||Meat, Food	15.69
||Korean Food||Meat, Food||Grilled Pork	0.0
||Bar||Pub	0.91 (0.3*)
...	..
||Korean Snack[8]	50 (42.3*)
...	..
Restaurant	100
||Night meal	71.43

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"*.

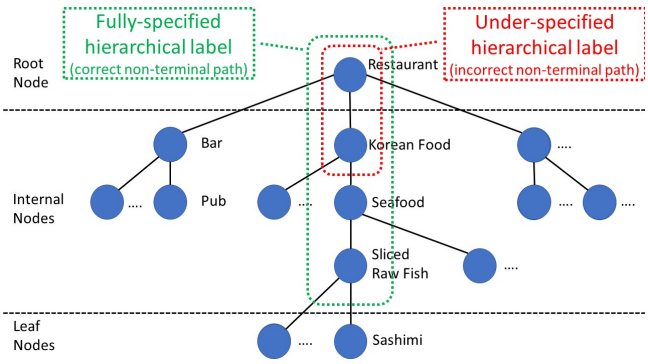


Figure 3: Example of under-specified and fully-specified hierarchical labels. Notice that a fully-specified hierarchical label can correspond to a correct non-terminal path of categories i.e. a path that terminates at an internal node. Our objective is to design a classifier robust to under-specified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

Given T, y should ideally represent a fully-specified path of categories $t = (c^0, c^1, ..., c^m)$.[9] 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.

[9]We omit subscripts for clarity.

5 OUR APPROACH

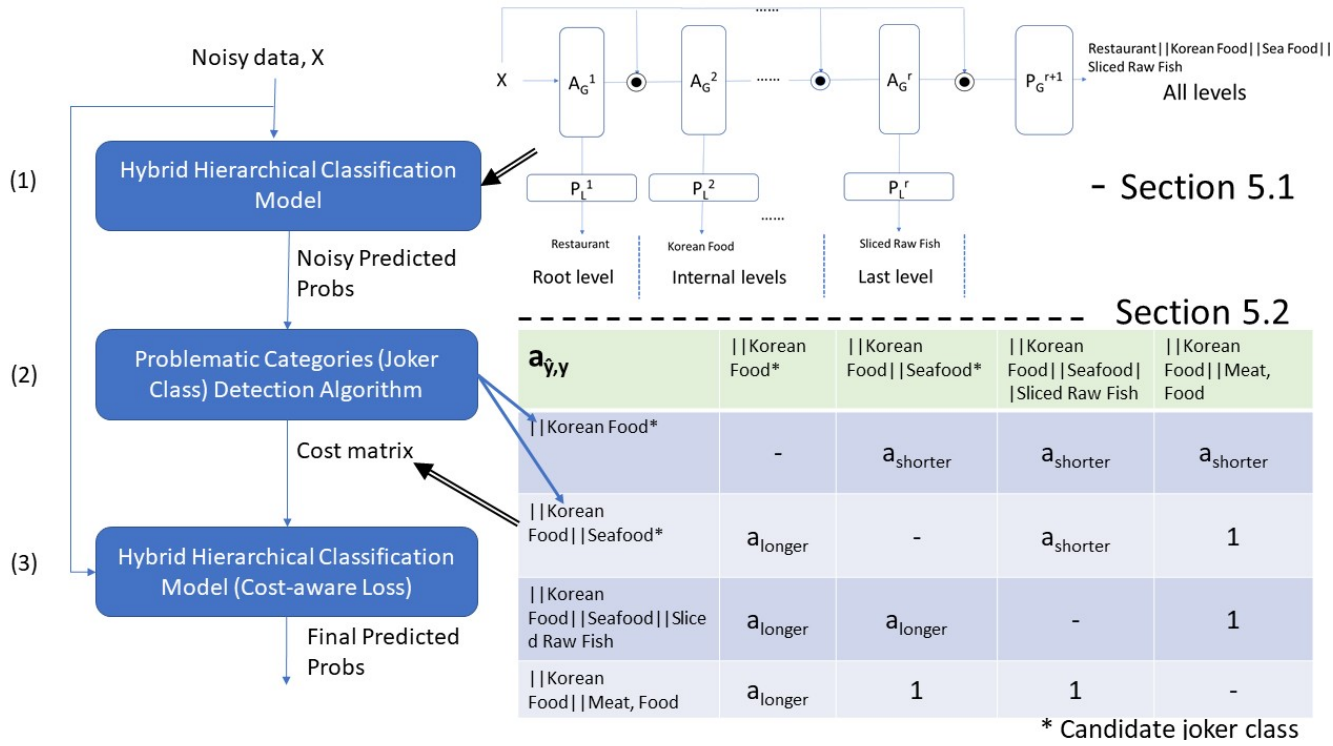
To allow robust learning in the presence of annotations with under-specified hierarchical labels, we propose the approach illustrated in Figure 4: (1). We develop a hybrid hierarchical classifier that combines one global and potentially several local classifiers (c.f. Section 5.1) using standard categorical 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), (3). We introduce a cost-based weight to 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.

5.1 Hybrid hierarchical classification model

Wehrmann et al. [25] has shown that a hierarchical classifier that performs both local and global optimisation, has significant advantages over adopting one of the two approaches only.

Following Wehrmann et al. [25], 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 $\mathcal{L}_L^h$, and one global output for the full category path, with a global loss $\mathcal{L}_G$. 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. [14], 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. [25] 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 $\mathcal{L}_G$ and all local output losses $\mathcal{L} = \mathcal{L}_G + \sum_{h=1}^{r} \mathcal{L}_L^h$, where $r \leq k$. As we want the



$a_{\hat{y},y}$	||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	-

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 $a_{shorter}$.

classes to be mutually exclusive for each level, we use the standard categorical cross-entropy loss for each $\mathcal{L}_L^h$ and for $\mathcal{L}_G$.

Differently to Wehrmann et al. [25], 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[10]. Accordingly, we change $\mathcal{L}_G$ from the standard categorical cross-entropy to the following weighted-by-sample categorical cross-entropy loss.

$$\mathcal{L}'_G = \frac{1}{N}\sum_{i=1}^{N} a_{\hat{y}_i,y_i}\mathcal{L}_{G,i} \quad (1)$$

[10]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 Section 6.3.2.

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

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

$\hat{t}_i$ 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 concentrate 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 are over-punishing 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 introduced in the previous section 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 [3, 26]: 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 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 the d to 3 and s to 100. We set $pt > 0.9$ and rt to the median of all the categories remaining after the previous steps are applied, resulting in 22 candidate joker classes.

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

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. Details are provided below.

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. As an illustrative example, we can find 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.[11]

The dataset is heavily imbalanced in terms of the number of POI instances attributed to each category, as illustrated in Figure 5.

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. Table 4 shows the top 10 head categories and the percentage of POIs attributed to them in the dataset. These 10 categories account for 44% of all the POIs. [12,13]

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 10% for testing purposes. We would like to highlight here, that actually the test dataset is a silver standard, as labels are under-specified for a part of the POIs,

[11] The actual characteristics of our dataset are closer to previous work in item classification for online product listings [9].
[12] The root category *||Restaurant* is common to all classes, so we remove it for the interest if space.
[13] All categories are translated from the Korean original version in English.

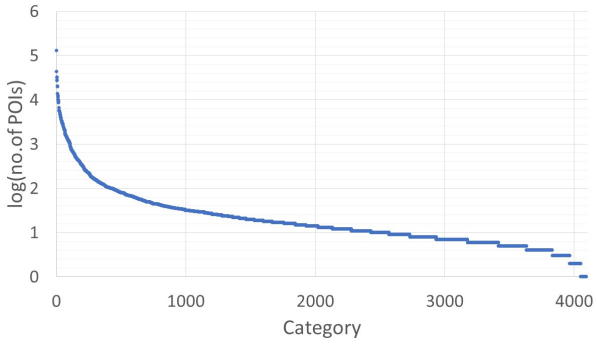


Figure 5: Our dataset is highly imbalanced.

Table 4: Percentage of POIs attributed to the top ten most popular categories in our dataset. These 10 categories account for 44% of all POIs.

Category	POI perc.
||Korean Food	15.61%
||Cafe, Dessert	5.33%
||Korean Food||Meat, Food	3.92%
||Cafe, Dessert||Cafe	3.86%
||Chinese Food||Chinese Restaurant	3.38%
||Bar||Beer,Hof,Pub||Beer,Hof	3.27%
||Chicken dakgangjeong	2.48%
||Korean Food||Seafood||Sliced Raw Fish||Sashimi	2.44%
||Korean Snack	2.4%
||Korean Food||Meat,Food||Pork||Grilled Pork	1.69%

as already discussed in previous sections. As already mentioned in Section 3, the gold standard includes 1000 POIs that we carefully verified. It is important to highlight again 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 [14]**: We use the flat classifier of Lagos et al. [14] 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 pre-defined 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 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 share the same hyper-parameter values.
- **Hcls**: Hcls stands for the hybrid hierarchical model described in Section 5.1. In Table 5, we report only the model that performs the best, which is the one that takes into account only the second level of the hierarchy. A detailed ablation study is included in Appendix B. After each dense layer that is added in order to represent 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 [5]. In that context, focal-loss [16], as well as its combination with class-balanced loss have shown the most promising results recently [6]. 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 [6]. 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 [16] and [6] for more details on these methods.
- **C$\mathcal{L}$**: C$\mathcal{L}$ 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$\mathcal{L}$**: CJ$\mathcal{L}$ corresponds to the joker-class specific loss introduced in Section 5.2.2. The suffixes indicate the cost values used. For instance, CJ$\mathcal{L}_{1.4,0.5}$, corresponds to the same cost values as the ones used for the C-$\mathcal{L}$ model, while CJ$\mathcal{L}_{5.0,0.5}$, means that the cost given to misclassifications with shorter paths is 5.0. A detailed ablation study showing how different values influence the performance of the model is presented in Section 6.3.2.

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 recommended in [11]. We use the standard macro and micro metrics for the evaluation calculated using the scikit-learn package [21].

6.3 Results

6.3.1 Overall results. Our models achieve the best results on the gold standard, as shown in Table 5. We observe an improvement of 2.87 points in micro-F1 compared to the baseline and 2.12 points

Table 5: 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$\mathcal{L}_{1.4,0.5}$ performs well on both the silver and gold standards (most balanced performance). Hcls+CJ$\mathcal{L}_{5.0,0.5}$ has the best overall performance on the gold standard. (⇑) denotes the delta in F1 between the two models above and the baseline on the gold standard, while (↑) denotes the delta in F1 to the hierarchical classifier, Hcls.

Dataset	Silver standard			Gold standard		
Model	Metric					
	Micro-prec.	Micro-rec.	Micro-F1	Micro-prec.	Micro-rec.	Micro-F1
Base [14]	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)
Hcls+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)
Hcls+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						
Hcls+C$\mathcal{L}$	73.27(±0.41)	62.90(±0.59)	67.69(±0.22)	52.67(±0.51)	31.94(±1.00)	39.76(±0.78)
Hcls+CJ$\mathcal{L}_{1.4,0.5}$	73.34(±0.4)	63.37(±0.27)	67.99(±0.17)	53.60(±1.17)	**33.51(±0.26)**	41.23(±0.4)
Hcls+CJ$\mathcal{L}_{1.4,-}$	**73.70(±0.17)**	62.97(±0.18)	67.91(±0.05)	53.67(±1.64)	32.31(±0.34)	40.34(±0.70)
Hcls+CJ$\mathcal{L}_{-,0.5}$	72.84(±0.30)	**63.93(±0.28)**	**68.09(±0.03)**	51.41(±1.71)	33.22(±0.91)	40.36(±1.01)
Hcls+CJ$\mathcal{L}_{5.0,0.5}$	73.58(±0.57)	59.09(±0.37)	65.54(±0.43)	**62.62(±0.99)**	33.04(±0.77)	**43.25(±0.56) (↑4.14)(⇑4.89)**
	Macro-prec.	Macro-rec.	Macro-F1	Macro-prec.	Macro-rec.	Macro-F1
Base [14]	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)
Hcls+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)
Hcls+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						
Hcls+C$\mathcal{L}$	80.50(±0.36)	82.89(±0.18)	80.51(±0.29)	50.60(±1.5)	40.87(±1.4)	43.08(±1.04)
Hcls+CJ$\mathcal{L}_{1.4,0.5}$	80.72(±0.2)	83.46(±0.29)	80.87(±0.22)	51.04(±0.92)	40.54(±0.51)	42.98(±0.41)
Hcls+CJ$\mathcal{L}_{1.4,-}$	80.65(±0.13)	83.10(±0.22)	80.67(±0.09)	**52.12(±0.77)**	41.48(±0.82)	**43.93(±0.74)**
Hcls+CJ$\mathcal{L}_{-,0.5}$	**80.76(±0.14)**	**83.50(±0.43)**	**80.90(±0.14)**	49.94(±0.67)	40.37(±0.21)	42.57(±0.36)
Hcls+CJ$\mathcal{L}_{5.0,0.5}$	80.34(±0.48)	83.37(±0.39)	80.55(±0.42)	51.04(±1.66)	**41.50(±1.45)**	43.80(±1.51) **(↑2.88) (⇑3.98)**

compared to the initial hierarchical model for the Hcls+CJ$\mathcal{L}_{1.4,0.5}$ model.

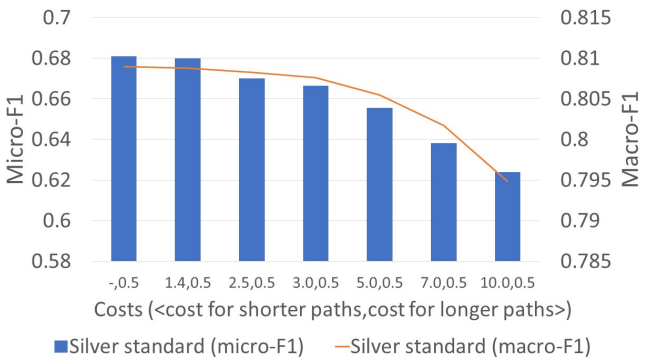
The improvement reaches 3.16 and 2.06 points respectively in macro-F1. Please note that the absolute scores may seem quite low, however, the gold standard consists exclusively of a subset of examples that the preliminary flat model fails to classify correctly as per the silver dataset, thus being very difficult to categorise (c.f. Section 3). Hcls+CJ$\mathcal{L}_{1.4,0.5}$ performs better than the class-agnostic model, *Hcls+C*$\mathcal{L}$, in terms of micro-F1. However, *Hcls+C*$\mathcal{L}$ is comparable, if not slightly better (by 0.1%), in macro-F1. These results are not surprising. *Hcls+C*$\mathcal{L}$ applies the costs to all misclassifications, implicitly pushing the model to predict long-tail categories in a stronger manner than in the case of *Hcls+C*$\mathcal{JL}_{1.4,0.5}$. Because of that, more POIs from head categories[14], are wrongly misclassified, resulting in the drop in micro-F1. To some extent, the additional POIs attributed to long-tail categories[15], smooth out this difference in macro-F1. However, *Hcls+C*$\mathcal{JL}_{1.4,0.5}$ has better overall performance.

On the silver standard, the *Hcls+C*$\mathcal{JL}_{1.4,0.5}$ model achieves better results than the no-cost models in terms of macro-F1, gaining 2.15 points compared to the baseline and 0.79 points to the 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 to the *Hcls+C*$\mathcal{JL}_{1.4,0.5}$ model (0.09 points decrease). This is, once more, 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 *Hcls+C*$\mathcal{JL}_{1.4,0.5}$ model has significantly better scores than the *Hcls* one, is also a strong indication of that.

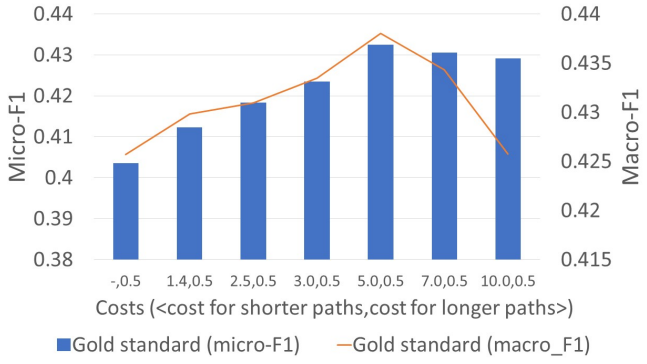
Surprisingly, the *Hcls+focal-cb* model does not have singificantly better scores when compared to the initial *Hcls* model on the gold standard. This is in contrast to combinations of the focal and class-based losses with the flat classifier model, which outperform the

[14]Most heavily populated categories found at the head, i.e. left-most part, of the data distribution in Figure 5.

[15]Less heavily populated categories.

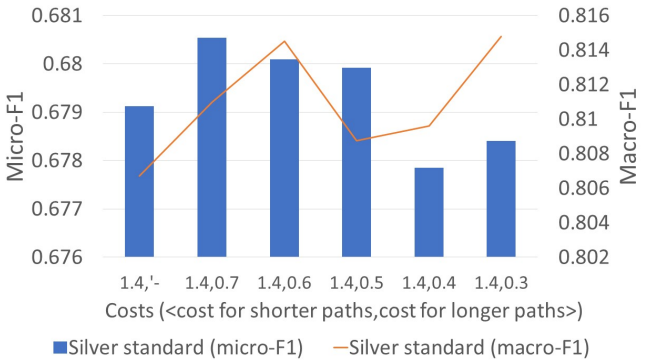


(a) Silver standard

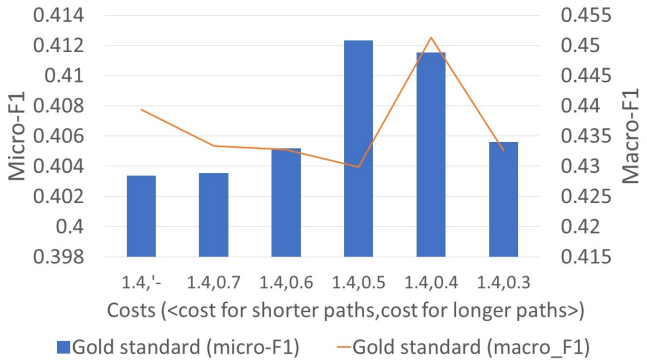


(b) Gold standard

Figure 6: F1 scores evolution as the cost values related to shorter path miclassifications change, while we keep the cost for longer path misclassifications fixed at 0.5. Both micro and macro-F1 scores increase up to the cost 5.0 for shorter path misclassifications in the gold standard.



(a) Silver standard



(b) Gold standard

Figure 7: F1 scores evolution as the cost values related to longer path miclassifications change, while we keep the cost for shorter path misclassifications fixed at 1.4. At a cost of 0.4 for longer paths, we observe a peak in both macro and micro-F1 in the gold standard, with the model predicting significantly more long-tail categories. However, the evolution is less stable than in the case of penalising shorter path misclassifications.

baseline, as expected, since they tend to favor long-tail rather than head categories, as shown in Appendix C. 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.

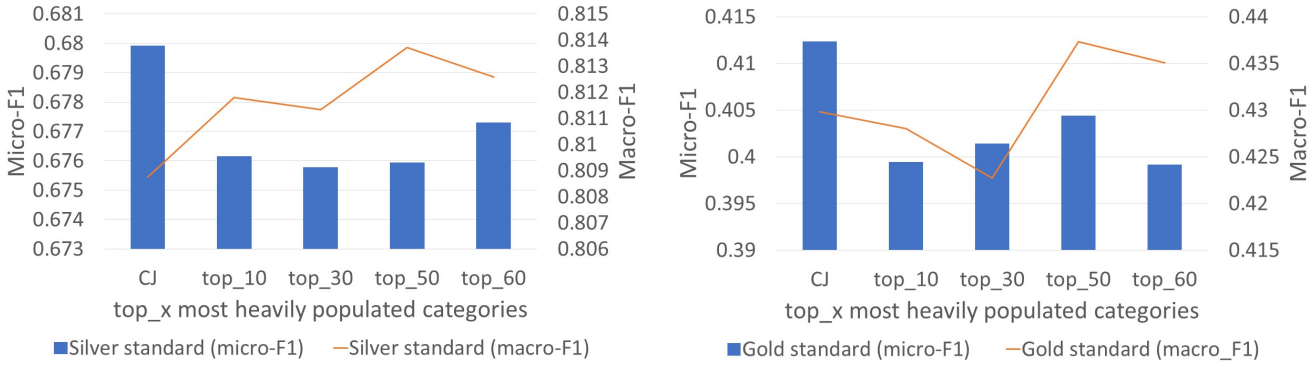
6.3.2 Ablation study. In this Section, we analyse two important aspects of our approach. The impact of misclassification costs and of selecting different sets of joker classes. All new results reported in this section are three-run averages.

Impact of misclassification costs. Figures 6 and 7 illustrate how the results change as we modulate the value of each of the two costs we have defined. We use the Hcls+CJ$\mathcal{L}$ model for both Figures.

Increasing the cost of misclassifications related to predicting shorter paths improves the results on the gold standard, reaching a maximum score at the value of 5.0 for both micro and macro-F1. The scores on the silver standard deteriorate, as the dataset has the same issue as the training data i.e. a skewed distribution because of joker classes. As the cost increases, more POIs are re-attributed from head to long-tail classes, causing the drop in the silver standard and the increase in the gold one.

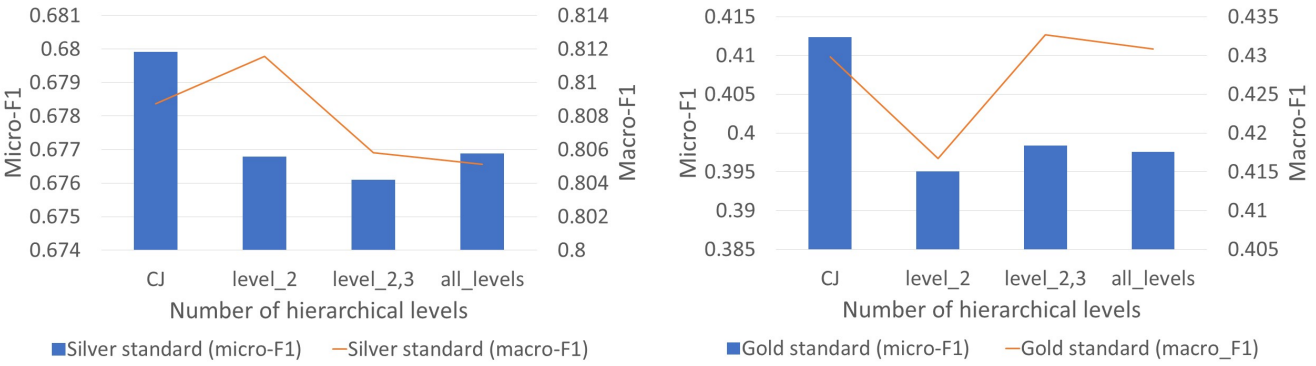
Decreasing the cost of misclassifications related to longer path predictions improves micro-F1 on the gold standard. Top values are reached at costs 0.5 and 0.4. Macro-F1 results are less stable. They decrease up to the cost of 0.5. However, at 0.4 there is a sudden peak, with the model predicting significantly more long-tail categories.

An important observation is that **both costs not only help, but they are actually rather complementary**. For instance, as shown in Table 5, **keeping only the cost related to shorter paths gives high precision scores in both datasets. On the other hand, micro-recall mainly benefits from the cost given**



(a) Silver standard **(b) Gold standard**

Figure 8: Comparison of the approach introduced in Section 5.2.2 for detecting joker classes (CJ) vs. selecting the top 10-60 head categories. CJ exhibits the most stable behaviour and performs consistently better in micro-F1. Penalising more than top-50 head categories achieves better macro-F1 than CJ, but lowers micro-F1 significantly.



(a) Silver standard **(b) Gold standard**

Figure 9: Comparison of the approach introduced in Section 5.2.2 for detecting joker classes (CJ) vs. selecting categories from the top hierarchy levels. CJ exhibits the most stable behaviour and performs consistently better in micro-F1.

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 the costs <5.0,0.5> results in the best model (best balance of micro and macro-F1 scores) on the gold standard. The model gains 4.89 points on the micro-F1 when compared to the baseline and 4.14 points compared to the hierarchical model. The improvement reaches respectively 3.98% and 2.88% in macro-F1.

Impact of joker classes. We have already observed in the results of Table 5 that selecting a set of joker classes rather than applying the costs in a class-agnostic manner, results in better micro-F1 and more balanced overall performance. However, as a direct implication of our definition, it is natural to assume that most of joker classes should (i) be part of the head categories and (ii) be located at the top levels of the categorisation hierarchy.

In Figure 8, we demonstrate how the scores change according to the number of head categories considered as joker classes. We apply costs <1.4,0.5> to misclassifications in all experiments. As we can see, the macro-F1 increases as more head categories are penalised, reaching a maximum value when the top fifty head categories are considered. Interestingly, when compared to the set of categories detected using the joker-class algorithm (c.f. Section 5.2.2), the micro-F1 is significantly lower. We thus observe a similar behavior to the class-agnostic model. Macro-F1 is increased as the number POIs re-attributed to long-tail categories increase up to a threshold (top_50 in this case), where the number of re-attributions results in incorrect classifications and thus hurts the micro-F1 scores. **The joker-class detection algorithm exhibits the most stable behaviour overall, and is consistently the best approach in terms of micro-F1.**

In Figure 9, we show how the scores evolve as we increase the hierarchical levels from which we extract categories. In detail, 17 categories are taken from the second level and 903 categories from the third level. We observe that as more levels are added, the macro-F1 increases on the gold standard, while the micro-F1 remains relatively unchanged. Compared to using the technique of Section 5.2.2, micro-F1s are lower in both datasets, while the macro-F1 is slightly

higher in the gold standard after the third level of the hierarchy is added. These results are not surprising, as after the third level, the misclassification costs are applied to significantly more categories (22 categories for CJ and 920 for the $level_2, 3$ model). Thus, the model tends to predict long-tail categories in a stronger fashion. While this results in a small improvement in macro-F1, the decrease in terms of micro-F1 is significant. **Overall, the *Hcls+CJℒ* model exhibits the most balanced performance.**

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 (≈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 more 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. After extensive experiments we found 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.

Other real-world services based on large and deep category hierarchies, such as in the e-commerce domain, may also be prone to under-specified hierarchical labels. It is therefore important future work to explore this issue in other domains and characterise in more detail the limitations of using manually annotated data as ground truth in practical settings.

A BASELINE DETAILS

The flat classifier proposed by Lagos et al. [14] is our baseline. It encodes text using 1-gram character LSTMs. All reported results are computed on the development dataset (c.f. Section 6.1). We compare here our baseline with the fairseq standard transformer-based model [19]. We also use the byte-pair encoding (BPE) [22] pre-processing step for both models (we use the extension "_bpe in Table 6). The results of the two models are comparable. We report here only the results of the final optimal configuration for Lagos et al. [14], where most importantly predictions having a minimum probability threshold of 0.4 were counted only. Note that POI attributes in Naver's database are in their majority written in Korean[16]. We have used the NFC unicode normalization format.

Table 6: Fairseq standard transformer-based model [19] and the flat classifier proposed by Lagos et al. [14], in our setting.

Model	Micro-prec.	Micro-rec.	Micro-F1
Transformer	0.6511	0.6511	0.6511
Transformer_bpe	0.6599	0.6599	0.6599
Lagos et al. [14]	0.7034	0.6338	**0.6668**
	Macro-prec.	Macro-rec.	Macro-F1
Transformer	0.7561	0.7724	0.7518
Transformer_bpe	0.7469	0.7679	0.7445
Lagos et al. [14]	0.7619	0.7723	**0.7541**

B HIERARCHICAL CLASSIFIER DETAILS

Tables 7 and 8 show how scores evolve in our silver and gold standard, as more local classifier layers are added to our architecture, to take into account more hierarchical levels. Contrary to state-of-the-art results that improve as more hierarchy levels are added, in our case results deteriorate. We assume that this may be related to the issue of under-specified hierarchical labels. A more detailed study is required though to confirm this hypothesis.

Table 7: Performance of hierarchical model on the silver standard. The suffix indicates the levels of the hierarchy included in the model.

Dataset	Silver standard		
Model	Metric		
	Micro-prec.	Micro-rec.	Micro-F1
$Hcls_2$	72.85(±0.26)	63.90(±0.26)	68.08(±0.05)
$Hcls_{2,3}$	73.17(±0.16)	63.52(±0.26)	68.00(±0.1)
$Hcls_{2,3,4}$	73.10(±0.18)	63.48(±0.25)	67.95(±0.12)
	Macro-prec.	Macro-rec.	Macro-F1
$Hcls_2$	80.29(±0.18)	82.23(±0.51)	80.08(±0.32)
$Hcls_{2,3}$	78.69(±0.62)	80.66(±0.82)	78.44(±0.69)
$Hcls_{2,3,4}$	76.97(±0.86)	78.52(±0.95)	76.51(±0.87)

[16] Although some multi-script names can also be found.

Table 8: Performance of the hierarchical model on the gold standard. The suffix indicates the levels of the hierarchy included in the model.

Dataset	Gold standard		
Model	Micro-prec.	Micro-rec.	Micro-F1
$Hcls_2$	50.32(±1.14)	31.99(±0.46)	39.11(±0.58)
$Hcls_{2,3}$	49.92(±1.02)	31.74(±0.48)	38.80(±0.61)
$Hcls_{2,3,4}$	50.78(±0.82)	32.03(±1.49)	39.26(±1.19)
	Macro-prec.	Macro-rec.	Macro-F1
$Hcls_2$	49.66(±1.69)	38.38(±1.10)	40.92(±1.33)
$Hcls_{2,3}$	48.72(±1.84)	38.20(±1.09)	40.47(±1.43)
$Hcls_{2,3,4}$	48.19(±1.52)	37.46(±1.1)	39.78(±1.18)

Table 9: Average performance (%) over 5 runs on the silver standard of the flat classifier based models.

Dataset	Silver standard		
Model	Metric		
	Micro-prec.	Micro-rec.	Micro-F1
Base	70.41(±0.27)	63.78(±0.26)	66.93(±0.2)
Base+Focal	70.09(±0.16)	63.56(±0.17)	66.67(±0.08)
Base+Focal-cb	70.22(±0.21)	63.56(±0.16)	66.73(±0.07)
Base+C$\mathcal{L}$	71.15(±0.21)	62.88(±0.19)	66.76(±0.11)
Base+CJ$\mathcal{L}_{1.4,0.5}$	70.94(±0.36)	63.63(±0.13)	67.08(±0.13)
	Macro-prec.	Macro-rec.	Macro-F1
Base	79.18(±1.00)	80.38(±0.86)	78.72(±0.91)
Base+Focal	78.20(±0.68)	78.50(±0.76)	77.28(±0.72)
Base+Focal-cb	78.32(±0.24)	79.01(±0.48)	77.60(±0.36)
Base+C$\mathcal{L}$	80.00(±0.15)	81.62(±0.81)	79.70(±0.14)
Base+CJ$\mathcal{L}_{1.4,0.5}$	79.97(±0.31)	81.46(±0.63)	79.64(±0.45)

C FLAT CLASSIFIER MODELS

Tables 9 and Table 10 show the results of the flat classifier based models on the silver and gold standards. Contrary to the hierarchical case, the model that uses a custom loss to tackle the imbalance in the dataset, *Base+focal-cb*, performs better than the initial *Base* model. That happens especially on the gold standard, as one would expect.

Table 10: Average performance (%) over 5 runs on the gold standard of the flat classifier based models. Standard deviation is also reported.

Dataset	Gold standard		
Model	Metric		
	Micro-prec.	Micro-rec.	Micro-F1
Base	46.57(±2.02)	32.62(±0.47)	38.36(±0.88)
Base+Focal	46.77(±2.23)	32.69(±1.33)	38.48(±1.67)
Base+Focal-cb	47.67(±1.34)	33.42(±0.82)	39.29(±0.98)
Base+C$\mathcal{L}$	50.52(±0.74)	33.31(±0.7)	40.14(±0.45)
Base+CJ$\mathcal{L}_{1.4,0.5}$	49.76(±0.73)	33.8(±0.58)	40.25(±0.48)
	Macro-prec.	Macro-rec.	Macro-F1
Base	47.75(±1.6)	37.80(±1.39)	39.82(±1.33)
Base+Focal	46.91(±1.83)	36.82(±0.57)	39.15(±0.77)
Base+Focal-cb	47.80(±0.45)	38.13(±0.66)	40.26(±0.67)
Base+C$\mathcal{L}$	50.13(±1.7)	40.42(±1.62)	42.49(±1.60)
Base+CJ$\mathcal{L}_{1.4,0.5}$	48.38(±1.23)	40.26(±1.22)	41.99 (±1.23)

REFERENCES

[1] Aixin Sun and Ee-Peng Lim. 2001. Hierarchical text classification and evaluation. In *Proceedings 2001 IEEE International Conference on Data Mining*. 521–528. https://doi.org/10.1109/ICDM.2001.989560

[2] Rohit Babbar, Ioannis Partalas, Eric Gaussier, and Massih-Reza Amini. 2013. On Flat versus Hierarchical Classification in Large-Scale Taxonomies. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2* (Lake Tahoe, Nevada) *(NIPS'13)*. Curran Associates Inc., Red Hook, NY, USA, 1824–1832.

[3] Avrim Blum and Tom Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-Training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory* (Madison, Wisconsin, USA) *(COLT' 98)*. Association for Computing Machinery, New York, NY, USA, 92–100. https://doi.org/10.1145/279943.279962

[4] R. Cerri, Rodrigo C. Barros, A. Carvalho, and Y. Jin. 2016. Reduction strategies for hierarchical multi-label classification in protein function prediction. *BMC Bioinformatics* 17 (2016).

[5] P. Chu, Xiao Bian, Shaopeng Liu, and H. Ling. 2020. Feature Space Augmentation for Long-Tailed Data. In *ECCV*.

[6] Y. Cui, M. Jia, T. Lin, Y. Song, and S. Belongie. 2019. Class-Balanced Loss Based on Effective Number of Samples. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 9260–9269. https://doi.org/10.1109/CVPR.2019.00949

[7] Shou Feng, P. Fu, and Wenbin Zheng. 2018. A hierarchical multi-label classification method based on neural networks for gene function prediction. *Biotechnology Biotechnological Equipment* 32 (2018), 1613 – 1621.

[8] Eleonora Giunchiglia and Thomas Lukasiewicz. 2020. Coherent Hierarchical Multi-Label Classification Networks. *ArXiv* abs/2010.10151 (2020).

[9] Jung-Woo Ha, Hyuna Pyo, and Jeonghee Kim. 2016. 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* (San Francisco, California, USA). Association for Computing Machinery, New York, NY, USA, 107–115. https://doi.org/10.1145/2939672.2939678

[10] Tieke He, Hongzhi Yin, Zhenyu Chen, Xiaofang Zhou, Shazia Sadiq, and Bin Luo. 2016. A Spatial-Temporal Topic Model for the Semantic Annotation of POIs in LBSNs. *ACM Trans. Intell. Syst. Technol.* 8, 1, Article 12 (July 2016), 24 pages. https://doi.org/10.1145/2905373

[11] Diederik Kingma and Jimmy Ba. 2015. Adam: a method for stochastic optimization (2014). *arXiv preprint arXiv:1412.6980* 15 (2015).

[12] John Krumm and Dany Rouhana. 2013. Placer: Semantic Place Labels from Diary Data. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Zurich, Switzerland) *(UbiComp '13)*. ACM, New York, NY, USA, 163–172. https://doi.org/10.1145/2493432.2493504

[13] Maxat Kulmanov, M. A. Khan, and R. Hoehndorf. 2018. DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics* 34 (2018), 660 – 668.

[14] Nikolaos Lagos, Salah Ait-Mokhtar, and Ioan Calapodescu. 2020. Point-Of-Interest Semantic Tag Completion in a Global Crowdsourced Search-and-Discovery

Database. In *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) (Frontiers in Artificial Intelligence and Applications, Vol. 325)*, Giuseppe De Giacomo, Alejandro Catalá, Bistra Dilkina, Michela Milano, Senén Barro, Alberto Bugarín, and Jérôme Lang (Eds.). IOS Press, 2993–3000. https://doi.org/10.3233/FAIA200474

[15] Nikolaos Lagos, Salah Ait-Mokhtar, and Ioan Calapodescu. 2020. Point-Of-Interest Semantic Tag Completion in a Global Crowdsourced Search-and-Discovery Database. In *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI 2020)*. IOS Press, forthcoming.

[16] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. 2017. Focal Loss for Dense Object Detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 2999–3007. https://doi.org/10.1109/ICCV.2017.324

[17] Shaopeng Liu, Jifan Yu, Juanzi Li, and Lei Hou. 2020. Geographical Information Enhanced POI Hierarchical Classification. In *Web Information Systems and Applications*, Guojun Wang, Xuemin Lin, James Hendler, Wei Song, Zhuoming Xu, and Genggeng Liu (Eds.). Springer International Publishing, Cham, 108–119.

[18] Luca Masera and Enrico Blanzieri. 2019. AWX: An Integrated Approach to Hierarchical-Multilabel Classification. In *Machine Learning and Knowledge Discovery in Databases*, Michele Berlingerio, Francesco Bonchi, Thomas Gärtner, Neil Hurley, and Georgiana Ifrim (Eds.). Springer International Publishing, 322–336.

[19] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

[20] Sungjoon Park, Jeongmin Byun, Sion Baek, Yongseok Cho, and Alice Oh. 2018. Subword-level Word Vector Representations for Korean. In *Proceedings of the 56th Annual Meeting of the ACL*. 2429–2438.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[22] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 1715–1725. https://doi.org/10.18653/v1/P16-1162

[23] Carlos N. Jr. Silla and Ale xA. Freitas. 2011. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery* 22, 1-2 (2011), 31–72. https://doi.org/10.1007/s10618-010-0175-9

[24] Yan Wang, Zongxu Qin, Jun Pang, Yang Zhang, and Jin Xin. 2017. Semantic Annotation for Places in LBSN Through Graph Embedding. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (Singapore, Singapore) *(CIKM '17)*. ACM, New York, NY, USA, 2343–2346.

[25] Jonatas Wehrmann, Ricardo Cerri, and Rodrigo Barros. 2018. Hierarchical Multi-Label Classification Networks. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, Stockholmsmässan, Stockholm Sweden, 5075–5084. http://proceedings.mlr.press/v80/wehrmann18a.html

[26] David Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *33rd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Cambridge, Massachusetts, USA, 189–196. https://doi.org/10.3115/981658.981684

[27] Mao Ye, Dong Shou, Wang-Chien Lee, Peifeng Yin, and Krzysztof Janowicz. 2011. 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* (San Diego, California, USA) *(KDD '11)*. ACM, New York, NY, USA, 520–528. https://doi.org/10.1145/2020408.2020491

[28] Jingbo Zhou, Shan Gou, Renjun Hu, Dongxiang Zhang, Jin Xu, Xuehui Wu, Airong Jiang, and Hui Xiong. 2019. A Collaborative Learning Framework to Tag Refinement for Points of Interest. (2019). Accepted at KDD 2019.